

general\_ledger

Generated by Doxygen 1.8.1.2

Sat Jun 7 2014 17:56:02



# Contents

<b>1</b>	<b>Data Structure Index</b>	<b>1</b>
1.1	Data Structures . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Data Structure Documentation</b>	<b>7</b>
3.1	ds_list Struct Reference . . . . .	7
3.1.1	Detailed Description . . . . .	7
3.1.2	Field Documentation . . . . .	7
3.1.2.1	current . . . . .	8
3.1.2.2	data_destructor . . . . .	8
3.1.2.3	free_on_delete . . . . .	8
3.1.2.4	head . . . . .	8
3.1.2.5	length . . . . .	8
3.1.2.6	tail . . . . .	8
3.2	ds_list_element Struct Reference . . . . .	8
3.2.1	Detailed Description . . . . .	8
3.2.2	Field Documentation . . . . .	9
3.2.2.1	data . . . . .	9
3.2.2.2	next . . . . .	9
3.2.2.3	previous . . . . .	9
3.3	ds_map Struct Reference . . . . .	9
3.3.1	Detailed Description . . . . .	9
3.3.2	Field Documentation . . . . .	10
3.3.2.1	hash_size . . . . .	10
3.3.2.2	lists . . . . .	10
3.4	ds_map_str Struct Reference . . . . .	10
3.4.1	Detailed Description . . . . .	10
3.4.2	Field Documentation . . . . .	10
3.4.2.1	hash_size . . . . .	11
3.4.2.2	lists . . . . .	11

3.5	ds_record Struct Reference	11
3.5.1	Detailed Description	11
3.5.2	Field Documentation	11
3.5.2.1	fields	11
3.6	ds_recordset Struct Reference	12
3.6.1	Detailed Description	12
3.6.2	Field Documentation	12
3.6.2.1	field_lengths	12
3.6.2.2	headers	12
3.6.2.3	num_fields	12
3.6.2.4	records	13
3.7	ds_str Struct Reference	13
3.7.1	Detailed Description	13
3.7.2	Field Documentation	13
3.7.2.1	capacity	13
3.7.2.2	data	13
3.7.2.3	length	13
3.8	ds_vector Struct Reference	13
3.8.1	Detailed Description	13
3.8.2	Field Documentation	14
3.8.2.1	current	14
3.8.2.2	data	14
3.8.2.3	data_destructor	14
3.8.2.4	free_on_delete	14
3.8.2.5	size	14
3.9	kv_pair_node Struct Reference	14
3.9.1	Detailed Description	15
3.9.2	Field Documentation	15
3.9.2.1	key	15
3.9.2.2	key	15
3.9.2.3	next	15
3.9.2.4	value	15
3.9.2.5	value	15
3.10	params Struct Reference	16
<b>4</b>	<b>File Documentation</b>	<b>17</b>
4.1	lib/database/database.h File Reference	17
4.1.1	Detailed Description	18
4.2	lib/database/db_connection.h File Reference	18
4.2.1	Detailed Description	19

4.2.2	Function Documentation	19
4.2.2.1	db_connect	19
4.3	lib/database/db_entities.c File Reference	19
4.3.1	Detailed Description	20
4.3.2	Function Documentation	20
4.3.2.1	db_create_entities_table	20
4.3.2.2	db_drop_entities_table	20
4.3.2.3	db_list_entities_report	20
4.4	lib/database/db_entities.h File Reference	21
4.4.1	Detailed Description	21
4.4.2	Function Documentation	22
4.4.2.1	db_create_entities_table	22
4.4.2.2	db_drop_entities_table	22
4.4.2.3	db_list_entities_report	22
4.5	lib/database/db_internal.h File Reference	23
4.5.1	Detailed Description	23
4.6	lib/database/db_query.h File Reference	23
4.6.1	Detailed Description	24
4.6.2	Function Documentation	24
4.6.2.1	db_execute_query	24
4.7	lib/database/db_reporting.c File Reference	25
4.7.1	Detailed Description	25
4.7.2	Function Documentation	26
4.7.2.1	db_create_report_from_query	26
4.8	lib/database/db_reporting.h File Reference	26
4.8.1	Detailed Description	26
4.8.2	Function Documentation	26
4.8.2.1	db_create_recordset_from_query	26
4.8.2.2	db_create_report_from_query	27
4.9	lib/database/db_sampledata.c File Reference	27
4.9.1	Detailed Description	27
4.10	lib/database/db_sampledata.h File Reference	28
4.10.1	Detailed Description	28
4.11	lib/database/db_sql.h File Reference	29
4.11.1	Detailed Description	29
4.11.2	Function Documentation	29
4.11.2.1	db_create_entities_table_sql	29
4.11.2.2	db_create_users_table_sql	30
4.11.2.3	db_drop_entities_table_sql	30
4.11.2.4	db_drop_users_table_sql	30

4.11.2.5	<a href="#">db_list_entities_report_sql</a>	30
4.11.2.6	<a href="#">db_list_users_report_sql</a>	30
4.12	<a href="#">lib/database/db_structure.c File Reference</a>	30
4.12.1	Detailed Description	31
4.12.2	Function Documentation	31
4.12.2.1	<a href="#">db_create_database_structure</a>	31
4.12.2.2	<a href="#">db_delete_database_structure</a>	32
4.13	<a href="#">lib/database/db_structure.h File Reference</a>	32
4.13.1	Detailed Description	32
4.13.2	Function Documentation	33
4.13.2.1	<a href="#">db_create_database_structure</a>	33
4.13.2.2	<a href="#">db_delete_database_structure</a>	33
4.14	<a href="#">lib/database/db_users.c File Reference</a>	33
4.14.1	Detailed Description	34
4.14.2	Function Documentation	34
4.14.2.1	<a href="#">db_create_users_table</a>	34
4.14.2.2	<a href="#">db_drop_users_table</a>	34
4.14.2.3	<a href="#">db_list_users_report</a>	34
4.15	<a href="#">lib/database/db_users.h File Reference</a>	35
4.15.1	Detailed Description	35
4.15.2	Function Documentation	36
4.15.2.1	<a href="#">db_create_users_table</a>	36
4.15.2.2	<a href="#">db_drop_users_table</a>	36
4.15.2.3	<a href="#">db_list_users_report</a>	36
4.16	<a href="#">lib/database/dummy/db_dummy_create_entities_table_sql.c File Reference</a>	36
4.16.1	Detailed Description	36
4.16.2	Function Documentation	37
4.16.2.1	<a href="#">db_create_entities_table_sql</a>	37
4.17	<a href="#">lib/database/dummy/db_dummy_create_users_table_sql.c File Reference</a>	37
4.17.1	Detailed Description	37
4.17.2	Function Documentation	37
4.17.2.1	<a href="#">db_create_users_table_sql</a>	37
4.18	<a href="#">lib/database/dummy/db_dummy_drop_entities_table_sql.c File Reference</a>	37
4.18.1	Detailed Description	38
4.18.2	Function Documentation	38
4.18.2.1	<a href="#">db_drop_entities_table_sql</a>	38
4.19	<a href="#">lib/database/dummy/db_dummy_drop_users_table_sql.c File Reference</a>	38
4.19.1	Detailed Description	38
4.19.2	Function Documentation	38
4.19.2.1	<a href="#">db_drop_users_table_sql</a>	38

4.20 lib/database/dummy/db_dummy_general.c File Reference . . . . .	39
4.20.1 Detailed Description . . . . .	39
4.20.2 Function Documentation . . . . .	40
4.20.2.1 db_connect . . . . .	40
4.20.2.2 db_create_recordset_from_query . . . . .	40
4.20.2.3 db_execute_query . . . . .	40
4.21 lib/database/dummy/db_dummy_list_entities_report_sql.c File Reference . . . . .	40
4.21.1 Detailed Description . . . . .	41
4.21.2 Function Documentation . . . . .	41
4.21.2.1 db_list_entities_report_sql . . . . .	41
4.22 lib/database/dummy/db_dummy_list_users_report_sql.c File Reference . . . . .	41
4.22.1 Detailed Description . . . . .	41
4.22.2 Function Documentation . . . . .	41
4.22.2.1 db_list_users_report_sql . . . . .	41
4.23 lib/database/mysql/db_mysql_create_entities_table_sql.c File Reference . . . . .	42
4.23.1 Detailed Description . . . . .	42
4.23.2 Function Documentation . . . . .	42
4.23.2.1 db_create_entities_table_sql . . . . .	42
4.24 lib/database/mysql/db_mysql_create_users_table_sql.c File Reference . . . . .	42
4.24.1 Detailed Description . . . . .	42
4.24.2 Function Documentation . . . . .	43
4.24.2.1 db_create_users_table_sql . . . . .	43
4.25 lib/database/mysql/db_mysql_drop_entities_table_sql.c File Reference . . . . .	43
4.25.1 Detailed Description . . . . .	43
4.25.2 Function Documentation . . . . .	43
4.25.2.1 db_drop_entities_table_sql . . . . .	43
4.26 lib/database/mysql/db_mysql_drop_users_table_sql.c File Reference . . . . .	43
4.26.1 Detailed Description . . . . .	44
4.26.2 Function Documentation . . . . .	44
4.26.2.1 db_drop_users_table_sql . . . . .	44
4.27 lib/database/mysql/db_mysql_general.c File Reference . . . . .	44
4.27.1 Detailed Description . . . . .	45
4.27.2 Function Documentation . . . . .	45
4.27.2.1 db_connect . . . . .	45
4.27.2.2 db_create_recordset_from_query . . . . .	45
4.27.2.3 db_execute_query . . . . .	46
4.27.3 Variable Documentation . . . . .	46
4.27.3.1 conn_mss . . . . .	46
4.27.3.2 main_mss . . . . .	46
4.28 lib/database/mysql/db_mysql_list_entities_report_sql.c File Reference . . . . .	46

4.28.1 Detailed Description . . . . .	46
4.28.2 Function Documentation . . . . .	46
4.28.2.1 db_list_entities_report_sql . . . . .	46
4.29 lib/database/mysql/db_mysql_list_users_report_sql.c File Reference . . . . .	47
4.29.1 Detailed Description . . . . .	47
4.29.2 Function Documentation . . . . .	47
4.29.2.1 db_list_users_report_sql . . . . .	47
4.30 lib/datastruct/data_structures.h File Reference . . . . .	47
4.30.1 Detailed Description . . . . .	48
4.31 lib/datastruct/ds_list.c File Reference . . . . .	48
4.31.1 Detailed Description . . . . .	49
4.31.2 Function Documentation . . . . .	50
4.31.2.1 ds_list_create . . . . .	50
4.31.2.2 ds_list_destructor . . . . .	50
4.32 lib/datastruct/ds_list.h File Reference . . . . .	50
4.32.1 Detailed Description . . . . .	52
4.32.2 Typedef Documentation . . . . .	52
4.32.2.1 ds_list . . . . .	52
4.32.3 Function Documentation . . . . .	52
4.32.3.1 ds_list_append . . . . .	52
4.32.3.2 ds_list_create . . . . .	52
4.32.3.3 ds_list_destroy . . . . .	53
4.32.3.4 ds_list_destructor . . . . .	53
4.32.3.5 ds_list_element . . . . .	53
4.32.3.6 ds_list_get_next_data . . . . .	53
4.32.3.7 ds_list_get_prev_data . . . . .	54
4.32.3.8 ds_list_is_empty . . . . .	54
4.32.3.9 ds_list_length . . . . .	54
4.32.3.10 ds_list_remove_all . . . . .	54
4.32.3.11 ds_list_remove_tail . . . . .	55
4.32.3.12 ds_list_seek_end . . . . .	55
4.32.3.13 ds_list_seek_start . . . . .	55
4.33 lib/datastruct/ds_map.c File Reference . . . . .	55
4.33.1 Detailed Description . . . . .	56
4.33.2 Function Documentation . . . . .	57
4.33.2.1 ds_map_init . . . . .	57
4.33.2.2 ds_map_print_all . . . . .	57
4.34 lib/datastruct/ds_map.h File Reference . . . . .	57
4.34.1 Detailed Description . . . . .	58
4.34.2 Typedef Documentation . . . . .	58



4.34.2.1	<a href="#">ds_map</a>	58
4.34.3	<a href="#">Function Documentation</a>	58
4.34.3.1	<a href="#">ds_map_destroy</a>	58
4.34.3.2	<a href="#">ds_map_get_value</a>	58
4.34.3.3	<a href="#">ds_map_init</a>	59
4.34.3.4	<a href="#">ds_map_insert</a>	59
4.34.3.5	<a href="#">ds_map_print_all</a>	59
4.35	<a href="#">lib/datastruct/ds_map_str.c File Reference</a>	59
4.35.1	<a href="#">Detailed Description</a>	60
4.35.2	<a href="#">Function Documentation</a>	61
4.35.2.1	<a href="#">ds_map_str_init</a>	61
4.36	<a href="#">lib/datastruct/ds_map_str.h File Reference</a>	61
4.36.1	<a href="#">Detailed Description</a>	62
4.36.2	<a href="#">Typedef Documentation</a>	62
4.36.2.1	<a href="#">ds_map_str</a>	62
4.36.3	<a href="#">Function Documentation</a>	62
4.36.3.1	<a href="#">ds_map_str_destroy</a>	62
4.36.3.2	<a href="#">ds_map_str_get_value</a>	62
4.36.3.3	<a href="#">ds_map_str_init</a>	63
4.36.3.4	<a href="#">ds_map_str_insert</a>	63
4.37	<a href="#">lib/datastruct/ds_record.c File Reference</a>	63
4.37.1	<a href="#">Detailed Description</a>	64
4.37.2	<a href="#">Function Documentation</a>	64
4.37.2.1	<a href="#">ds_record_create</a>	64
4.37.2.2	<a href="#">ds_record_destructor</a>	65
4.37.2.3	<a href="#">ds_record_make_delim_string</a>	65
4.37.2.4	<a href="#">ds_record_make_values_string</a>	65
4.37.2.5	<a href="#">ds_record_tokenize</a>	65
4.38	<a href="#">lib/datastruct/ds_record.h File Reference</a>	65
4.38.1	<a href="#">Detailed Description</a>	67
4.38.2	<a href="#">Typedef Documentation</a>	67
4.38.2.1	<a href="#">ds_record</a>	67
4.38.3	<a href="#">Function Documentation</a>	67
4.38.3.1	<a href="#">ds_record_clear</a>	67
4.38.3.2	<a href="#">ds_record_create</a>	67
4.38.3.3	<a href="#">ds_record_destroy</a>	68
4.38.3.4	<a href="#">ds_record_destructor</a>	68
4.38.3.5	<a href="#">ds_record_get_field</a>	68
4.38.3.6	<a href="#">ds_record_get_next_data</a>	68
4.38.3.7	<a href="#">ds_record_make_delim_string</a>	68

4.38.3.8	<a href="#">ds_record_make_values_string</a>	69
4.38.3.9	<a href="#">ds_record_seek_start</a>	69
4.38.3.10	<a href="#">ds_record_set_field</a>	69
4.38.3.11	<a href="#">ds_record_size</a>	69
4.38.3.12	<a href="#">ds_record_tokenize</a>	69
4.39	<a href="#">lib/datastruct/ds_recordset.c File Reference</a>	70
4.39.1	<a href="#">Detailed Description</a>	70
4.39.2	<a href="#">Function Documentation</a>	71
4.39.2.1	<a href="#">ds_recordset_create</a>	71
4.40	<a href="#">lib/datastruct/ds_recordset.h File Reference</a>	71
4.40.1	<a href="#">Detailed Description</a>	72
4.40.2	<a href="#">Typedef Documentation</a>	73
4.40.2.1	<a href="#">ds_recordset</a>	73
4.40.3	<a href="#">Function Documentation</a>	73
4.40.3.1	<a href="#">ds_recordset_add_record</a>	73
4.40.3.2	<a href="#">ds_recordset_create</a>	73
4.40.3.3	<a href="#">ds_recordset_destroy</a>	73
4.40.3.4	<a href="#">ds_recordset_get_next_insert_query</a>	73
4.40.3.5	<a href="#">ds_recordset_get_text_report</a>	74
4.40.3.6	<a href="#">ds_recordset_next_record</a>	74
4.40.3.7	<a href="#">ds_recordset_num_fields</a>	74
4.40.3.8	<a href="#">ds_recordset_num_records</a>	74
4.40.3.9	<a href="#">ds_recordset_seek_start</a>	75
4.40.3.10	<a href="#">ds_recordset_set_headers</a>	75
4.41	<a href="#">lib/datastruct/ds_str.c File Reference</a>	75
4.41.1	<a href="#">Detailed Description</a>	77
4.41.2	<a href="#">Function Documentation</a>	77
4.41.2.1	<a href="#">ds_str_assign</a>	77
4.41.2.2	<a href="#">ds_str_assign_cstr</a>	77
4.41.2.3	<a href="#">ds_str_char_at_index</a>	78
4.41.2.4	<a href="#">ds_str_clear</a>	78
4.41.2.5	<a href="#">ds_str_compare</a>	78
4.41.2.6	<a href="#">ds_str_compare_cstr</a>	78
4.41.2.7	<a href="#">ds_str_concat</a>	78
4.41.2.8	<a href="#">ds_str_create</a>	79
4.41.2.9	<a href="#">ds_str_create_direct</a>	79
4.41.2.10	<a href="#">ds_str_create_sprintf</a>	79
4.41.2.11	<a href="#">ds_str_cstr</a>	80
4.41.2.12	<a href="#">ds_str_decorate</a>	80
4.41.2.13	<a href="#">ds_str_destroy</a>	80

4.41.2.14 ds_str_destructor . . . . .	80
4.41.2.15 ds_str_doubleval . . . . .	80
4.41.2.16 ds_str_dup . . . . .	81
4.41.2.17 ds_str_getline . . . . .	81
4.41.2.18 ds_str_intval . . . . .	81
4.41.2.19 ds_str_is_empty . . . . .	81
4.41.2.20 ds_str_length . . . . .	82
4.41.2.21 ds_str_split . . . . .	82
4.41.2.22 ds_str_strchr . . . . .	82
4.41.2.23 ds_str_substr_left . . . . .	82
4.41.2.24 ds_str_substr_right . . . . .	83
4.41.2.25 ds_str_trim . . . . .	83
4.41.2.26 ds_str_trim_leading . . . . .	83
4.41.2.27 ds_str_trim_trailing . . . . .	83
4.41.2.28 ds_str_trunc . . . . .	83
4.42 lib/datastruct/ds_str.h File Reference . . . . .	84
4.42.1 Detailed Description . . . . .	86
4.42.2 Typedef Documentation . . . . .	86
4.42.2.1 ds_str . . . . .	86
4.42.3 Function Documentation . . . . .	86
4.42.3.1 ds_str_assign . . . . .	86
4.42.3.2 ds_str_assign_cstr . . . . .	86
4.42.3.3 ds_str_char_at_index . . . . .	86
4.42.3.4 ds_str_clear . . . . .	87
4.42.3.5 ds_str_compare . . . . .	87
4.42.3.6 ds_str_compare_cstr . . . . .	87
4.42.3.7 ds_str_concat . . . . .	87
4.42.3.8 ds_str_concat_cstr . . . . .	88
4.42.3.9 ds_str_create . . . . .	88
4.42.3.10 ds_str_create_direct . . . . .	88
4.42.3.11 ds_str_create_sprintf . . . . .	88
4.42.3.12 ds_str_cstr . . . . .	89
4.42.3.13 ds_str_decorate . . . . .	89
4.42.3.14 ds_str_destroy . . . . .	89
4.42.3.15 ds_str_destructor . . . . .	89
4.42.3.16 ds_str_doubleval . . . . .	89
4.42.3.17 ds_str_dup . . . . .	90
4.42.3.18 ds_str_getline . . . . .	90
4.42.3.19 ds_str_hash . . . . .	90
4.42.3.20 ds_str_intval . . . . .	90

4.42.3.21 ds_str_is_empty . . . . .	91
4.42.3.22 ds_str_length . . . . .	91
4.42.3.23 ds_str_split . . . . .	91
4.42.3.24 ds_str_strchr . . . . .	91
4.42.3.25 ds_str_substr_left . . . . .	92
4.42.3.26 ds_str_substr_right . . . . .	92
4.42.3.27 ds_str_trim . . . . .	92
4.42.3.28 ds_str_trim_leading . . . . .	92
4.42.3.29 ds_str_trim_trailing . . . . .	93
4.42.3.30 ds_str_trunc . . . . .	93
4.43 lib/datastruct/ds_vector.c File Reference . . . . .	93
4.43.1 Detailed Description . . . . .	94
4.43.2 Function Documentation . . . . .	94
4.43.2.1 ds_vector_create . . . . .	94
4.43.2.2 ds_vector_destructor . . . . .	95
4.44 lib/datastruct/ds_vector.h File Reference . . . . .	95
4.44.1 Detailed Description . . . . .	96
4.44.2 Typedef Documentation . . . . .	96
4.44.2.1 ds_vector . . . . .	96
4.44.3 Function Documentation . . . . .	96
4.44.3.1 ds_vector_clear . . . . .	96
4.44.3.2 ds_vector_create . . . . .	96
4.44.3.3 ds_vector_destroy . . . . .	97
4.44.3.4 ds_vector_destructor . . . . .	97
4.44.3.5 ds_vector_element . . . . .	97
4.44.3.6 ds_vector_get_next_data . . . . .	97
4.44.3.7 ds_vector_seek_start . . . . .	98
4.44.3.8 ds_vector_set . . . . .	98
4.44.3.9 ds_vector_size . . . . .	98
4.45 lib/file_ops/config_file_read.c File Reference . . . . .	98
4.45.1 Detailed Description . . . . .	99
4.45.2 Macro Definition Documentation . . . . .	99
4.45.2.1 CONFIG_MAP_SIZE . . . . .	99
4.45.2.2 MAX_BUFFER_SIZE . . . . .	100
4.45.3 Function Documentation . . . . .	100
4.45.3.1 config_file_free . . . . .	100
4.45.3.2 config_file_read . . . . .	100
4.45.3.3 config_file_value . . . . .	100
4.46 lib/file_ops/config_file_read.h File Reference . . . . .	100
4.46.1 Detailed Description . . . . .	102

4.46.2	Macro Definition Documentation	102
4.46.2.1	CONFIG_FILE_MALFORMED_FILE	102
4.46.2.2	CONFIG_FILE_NO_FILE	102
4.46.2.3	CONFIG_FILE_OK	102
4.46.3	Function Documentation	102
4.46.3.1	config_file_free	102
4.46.3.2	config_file_read	102
4.46.3.3	config_file_value	103
4.47	lib/file_ops/delim_file_read.c File Reference	103
4.47.1	Detailed Description	104
4.47.2	Macro Definition Documentation	104
4.47.2.1	MAX_LINE_SIZE	104
4.47.3	Function Documentation	104
4.47.3.1	delim_file_read	104
4.48	lib/file_ops/delim_file_read.h File Reference	104
4.48.1	Detailed Description	106
4.48.2	Function Documentation	106
4.48.2.1	delim_file_read	106
4.49	lib/file_ops/file_ops.h File Reference	106
4.49.1	Detailed Description	107
4.50	lib/gl_general/gl_logging.c File Reference	108
4.50.1	Detailed Description	108
4.50.2	Function Documentation	108
4.50.2.1	gl_log_msg	108
4.50.2.2	gl_set_logging	109
4.51	lib/gl_general/gl_logging.h File Reference	109
4.51.1	Detailed Description	110
4.51.2	Function Documentation	110
4.51.2.1	gl_log_msg	110
4.51.2.2	gl_set_logging	110
4.52	main.c File Reference	110
4.52.1	Detailed Description	111
4.52.2	Function Documentation	111
4.52.2.1	main	111



# Chapter 1

## Data Structure Index

### 1.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">ds_list</a>	7
<a href="#">ds_list_element</a>	8
<a href="#">ds_map</a>	9
<a href="#">ds_map_str</a>	10
<a href="#">ds_record</a>	11
<a href="#">ds_recordset</a>	12
<a href="#">ds_str</a>	13
<a href="#">ds_vector</a>	13
<a href="#">kv_pair_node</a>	14
<a href="#">params</a>	16





## Chapter 2

# File Index

### 2.1 File List

Here is a list of all documented files with brief descriptions:

<b>config.h</b>	??
<b>main.c</b>	
Main function for general_ledger	110
lib/database/database.h	
User interface to database functionality	17
lib/database/db_connection.h	
Interface to database connection functionality	18
lib/database/db_entities.c	
Implementation of entities functionality	19
lib/database/db_entities.h	
Interface to entities functionality	21
lib/database/db_internal.h	
Internal library interface to database functionality	23
lib/database/db_query.h	
Interface to database query functionality	23
lib/database/db_reporting.c	
Implementation of database reporting functionality	25
lib/database/db_reporting.h	
Interface to database reporting functionality	26
lib/database/db_sampledata.c	
Implementation of database sample data functionality	27
lib/database/db_sampledata.h	
Interface to database sample data functionality	28
lib/database/db_sql.h	
Interface to database specific SQL strings	29
lib/database/db_structure.c	
Implementation of database structure functionality	30
lib/database/db_structure.h	
Interface to database structure functionality	32
lib/database/db_users.c	
Implementation of users functionality	33
lib/database/db_users.h	
Interface to users functionality	35
lib/database/dummy/db_dummy_create_entities_table_sql.c	
Returns dummy SQL query to create entities table	36
lib/database/dummy/db_dummy_create_users_table_sql.c	
Returns dummy SQL query to create users table	37

lib/database/dummy/db_dummy_drop_entities_table_sql.c	Returns dummy SQL query to drop entities table . . . . .	37
lib/database/dummy/db_dummy_drop_users_table_sql.c	Returns dummy SQL query to drop users table . . . . .	38
lib/database/dummy/db_dummy_general.c	Implementation of dummy database functionality . . . . .	39
lib/database/dummy/db_dummy_list_entities_report_sql.c	Returns dummy SQL query to create list entities report . . . . .	40
lib/database/dummy/db_dummy_list_users_report_sql.c	Returns dummy SQL query to create list users report . . . . .	41
lib/database/mysql/db_mysql_create_entities_table_sql.c	Returns MYSQL SQL query to create entities table . . . . .	42
lib/database/mysql/db_mysql_create_users_table_sql.c	Returns MYSQL SQL query to create users table . . . . .	42
lib/database/mysql/db_mysql_drop_entities_table_sql.c	Returns MYSQL SQL query to drop entities table . . . . .	43
lib/database/mysql/db_mysql_drop_users_table_sql.c	Returns MYSQL SQL query to drop users table . . . . .	43
lib/database/mysql/db_mysql_general.c	Implementation of MYSQL database functionality . . . . .	44
lib/database/mysql/db_mysql_list_entities_report_sql.c	Returns MYSQL SQL query to create list entities report . . . . .	46
lib/database/mysql/db_mysql_list_users_report_sql.c	Returns MYSQL SQL query to create list users report . . . . .	47
lib/datastruct/data_structures.h	Interface to data structures . . . . .	47
lib/datastruct/ds_list.c	Implementation of generic doubly-linked list data structure . . . . .	48
lib/datastruct/ds_list.h	Interface to generic doubly-linked list data structure . . . . .	50
lib/datastruct/ds_map.c	Implementation of string-string hash map data structure . . . . .	55
lib/datastruct/ds_map.h	Interface to string-string hash map data structure . . . . .	57
lib/datastruct/ds_map_str.c	Implementation of string-string hash map data structure . . . . .	59
lib/datastruct/ds_map_str.h	Interface to string-string hash map data structure . . . . .	61
lib/datastruct/ds_record.c	Implementation of record database structure . . . . .	63
lib/datastruct/ds_record.h	Interface to record data structure . . . . .	65
lib/datastruct/ds_recordset.c	Implementation of query result set structure . . . . .	70
lib/datastruct/ds_recordset.h	Interface to record set structure . . . . .	71
lib/datastruct/ds_str.c	Implementation of string data structure . . . . .	75
lib/datastruct/ds_str.h	Interface to string data structure . . . . .	84
lib/datastruct/ds_vector.c	Implementation of generic doubly-linked vector data structure . . . . .	93
lib/datastruct/ds_vector.h	Interface to generic doubly-linked vector data structure . . . . .	95
lib/file_ops/config_file_read.c	Implementation of configuration file reading functionality . . . . .	98
lib/file_ops/config_file_read.h	Interface to configuration file reading functionality . . . . .	100

lib/file_ops/ <a href="#">delim_file_read.c</a>	
Implementation of delimited file reading functionality . . . . .	103
lib/file_ops/ <a href="#">delim_file_read.h</a>	
Interface to delimited file reading functionality . . . . .	104
lib/file_ops/ <a href="#">file_ops.h</a>	
User interface to file operations functionality . . . . .	106
lib/gl_general/ <b>gl_errors.h</b> . . . . .	??
lib/gl_general/ <b>gl_general.h</b> . . . . .	??
lib/gl_general/ <a href="#">gl_logging.c</a>	
Implementation of logging functionality . . . . .	108
lib/gl_general/ <a href="#">gl_logging.h</a>	
Interface to logging functionality . . . . .	109

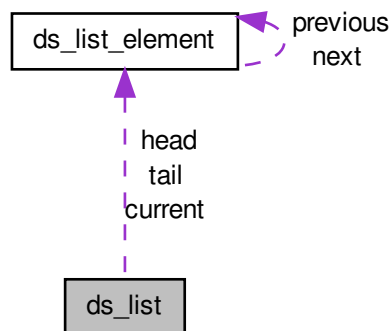


## Chapter 3

# Data Structure Documentation

### 3.1 ds\_list Struct Reference

Collaboration diagram for ds\_list:



#### Data Fields

- `size_t` `length`
- `bool` `free_on_delete`
- `struct ds_list_element *` `head`
- `struct ds_list_element *` `tail`
- `struct ds_list_element *` `current`
- `void(* data_destructor )(void *)`

#### 3.1.1 Detailed Description

List data structure

#### 3.1.2 Field Documentation

### 3.1.2.1 struct `ds_list_element*` `ds_list::current`

Pointer to current element

### 3.1.2.2 void(\* `ds_list::data_destructor`)(void \*)

Data destructor function

### 3.1.2.3 bool `ds_list::free_on_delete`

'Free on delete' flag

### 3.1.2.4 struct `ds_list_element*` `ds_list::head`

Pointer to head element

### 3.1.2.5 size\_t `ds_list::length`

Length of list

### 3.1.2.6 struct `ds_list_element*` `ds_list::tail`

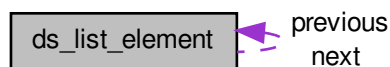
Pointer to tail element

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds\\_list.c](#)

## 3.2 ds\_list\_element Struct Reference

Collaboration diagram for `ds_list_element`:



### Data Fields

- void \* [data](#)
- struct [ds\\_list\\_element](#) \* [previous](#)
- struct [ds\\_list\\_element](#) \* [next](#)

### 3.2.1 Detailed Description

List element data structure

### 3.2.2 Field Documentation

#### 3.2.2.1 void\* ds\_list\_element::data

Pointer to data

#### 3.2.2.2 struct ds\_list\_element\* ds\_list\_element::next

Pointer to next element

#### 3.2.2.3 struct ds\_list\_element\* ds\_list\_element::previous

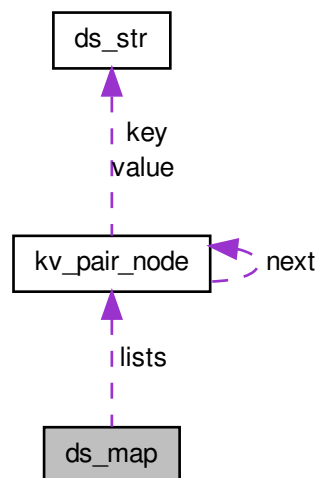
Pointer to previous element

The documentation for this struct was generated from the following file:

- lib/datastruct/[ds\\_list.c](#)

## 3.3 ds\_map Struct Reference

Collaboration diagram for ds\_map:



### Data Fields

- struct [kv\\_pair\\_node](#) \*\* `lists`
- size\_t [hash\\_size](#)

### 3.3.1 Detailed Description

Structure to hold a hash map

### 3.3.2 Field Documentation

#### 3.3.2.1 `size_t ds_map::hash_size`

Size of array of lists

#### 3.3.2.2 `struct kv_pair_node** ds_map::lists`

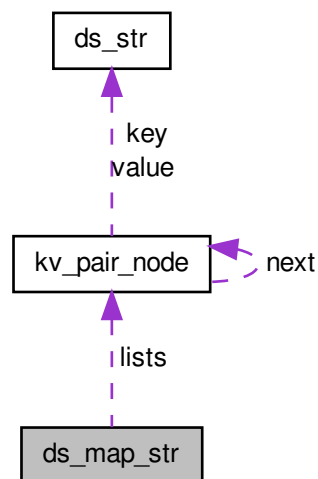
Pointer to array of lists

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds\\_map.c](#)

## 3.4 `ds_map_str` Struct Reference

Collaboration diagram for `ds_map_str`:



### Data Fields

- struct [kv\\_pair\\_node](#) \*\* `lists`
- `size_t` [hash\\_size](#)

#### 3.4.1 Detailed Description

Structure to hold a hash map

#### 3.4.2 Field Documentation



#### 3.4.2.1 `size_t ds_map_str::hash_size`

Size of array of lists

#### 3.4.2.2 `struct kv_pair_node** ds_map_str::lists`

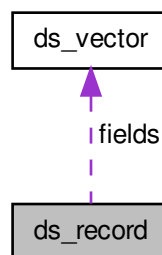
Pointer to array of lists

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds\\_map\\_str.c](#)

## 3.5 ds\_record Struct Reference

Collaboration diagram for ds\_record:



### Data Fields

- struct [ds\\_vector](#) \* `fields`

#### 3.5.1 Detailed Description

Vector data structure

#### 3.5.2 Field Documentation

##### 3.5.2.1 `struct ds_vector* ds_record::fields`

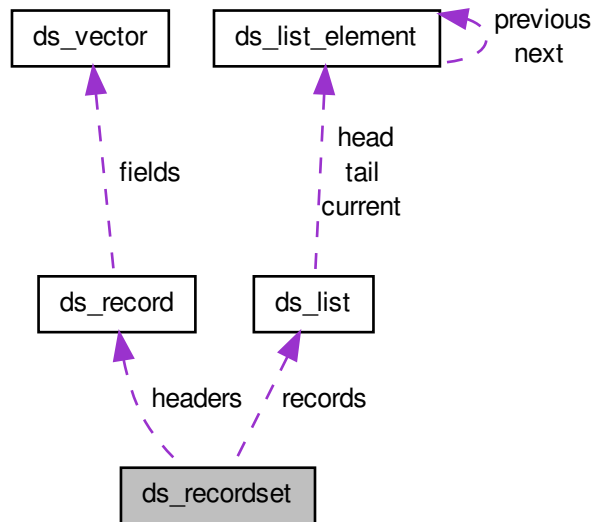
Vector of fields

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds\\_record.c](#)

### 3.6 ds\_recordset Struct Reference

Collaboration diagram for ds\_recordset:



#### Data Fields

- `size_t num_fields`
- `size_t * field_lengths`
- `ds_record headers`
- `ds_list records`

#### 3.6.1 Detailed Description

Result set structure

#### 3.6.2 Field Documentation

##### 3.6.2.1 `size_t * ds_recordset::field_lengths`

Lengths of the longest fields

##### 3.6.2.2 `ds_record ds_recordset::headers`

A list of field headers

##### 3.6.2.3 `size_t ds_recordset::num_fields`

The number of fields in a record

#### 3.6.2.4 ds\_list ds\_recordset::records

A list of records

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds\\_recordset.c](#)

## 3.7 ds\_str Struct Reference

### Data Fields

- `char *` [data](#)
- `size_t` [length](#)
- `size_t` [capacity](#)

#### 3.7.1 Detailed Description

Structure to contain string

#### 3.7.2 Field Documentation

##### 3.7.2.1 size\_t ds\_str::capacity

The size of the `data` buffer

##### 3.7.2.2 char\* ds\_str::data

The data in C-style string format

##### 3.7.2.3 size\_t ds\_str::length

The length of the string

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds\\_str.c](#)

## 3.8 ds\_vector Struct Reference

### Data Fields

- `size_t` [size](#)
- `size_t` [current](#)
- `bool` [free\\_on\\_delete](#)
- `void **` [data](#)
- `void(*` [data\\_destructor](#) `)(void *)`

#### 3.8.1 Detailed Description

Vector data structure

### 3.8.2 Field Documentation

#### 3.8.2.1 `size_t ds_vector::current`

Current position

#### 3.8.2.2 `void** ds_vector::data`

Data array

#### 3.8.2.3 `void(* ds_vector::data_destructor)(void *)`

Data destructor function

#### 3.8.2.4 `bool ds_vector::free_on_delete`

'Free on delete' flag

#### 3.8.2.5 `size_t ds_vector::size`

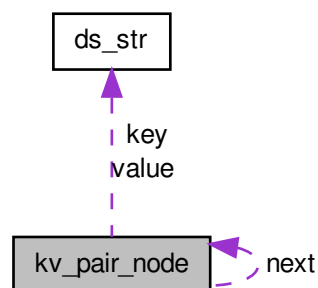
Size of vector

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds\\_vector.c](#)

## 3.9 kv\_pair\_node Struct Reference

Collaboration diagram for kv\_pair\_node:



### Data Fields

- `char * key`
- `char * value`
- `struct kv\_pair\_node * next`

- [ds\\_str key](#)
- [ds\\_str value](#)

### 3.9.1 Detailed Description

Structure to hold a key-value pair node

### 3.9.2 Field Documentation

#### 3.9.2.1 ds\_str kv\_pair\_node::key

A pointer to the key

#### 3.9.2.2 char\* kv\_pair\_node::key

A pointer to the key

#### 3.9.2.3 struct kv\_pair\_node \* kv\_pair\_node::next

A pointer to the next node

#### 3.9.2.4 ds\_str kv\_pair\_node::value

A pointer to the value

#### 3.9.2.5 char\* kv\_pair\_node::value

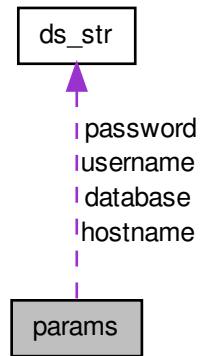
A pointer to the value

The documentation for this struct was generated from the following files:

- [lib/datastruct/ds\\_map.c](#)
- [lib/datastruct/ds\\_map\\_str.c](#)

### 3.10 params Struct Reference

Collaboration diagram for params:



#### Data Fields

- [ds\\_str](#) **hostname**
- [ds\\_str](#) **database**
- [ds\\_str](#) **username**
- [ds\\_str](#) **password**
- **bool help**
- **bool version**
- **bool create**
- **bool delete\_data**
- **bool sample**
- **bool list\_users**
- **bool list\_entities**

The documentation for this struct was generated from the following file:

- `config.h`

## Chapter 4

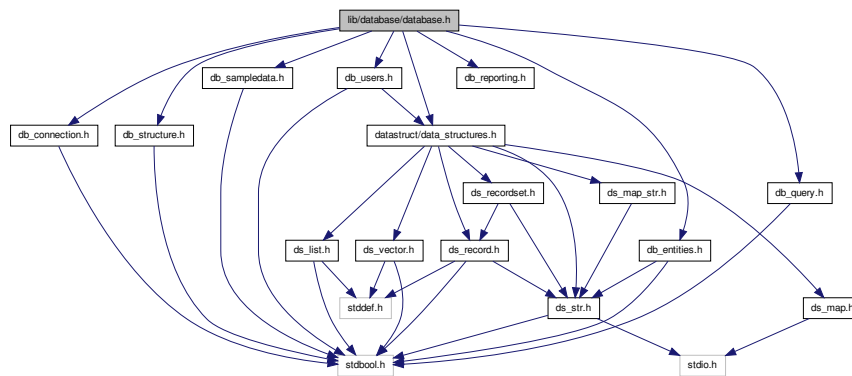
# File Documentation

### 4.1 lib/database/database.h File Reference

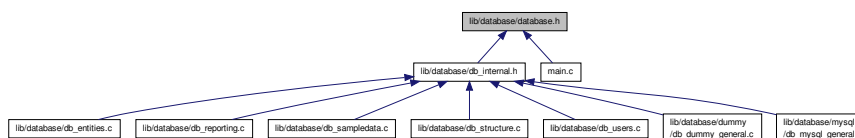
User interface to database functionality.

```
#include "datastruct/data_structures.h"
#include "db_connection.h"
#include "db_structure.h"
#include "db_query.h"
#include "db_sampledata.h"
#include "db_reporting.h"
#include "db_users.h"
#include "db_entities.h"
```

Include dependency graph for database.h:



This graph shows which files directly or indirectly include this file:



### 4.1.1 Detailed Description

User interface to database functionality.

#### Author

Paul Griffiths

#### Copyright

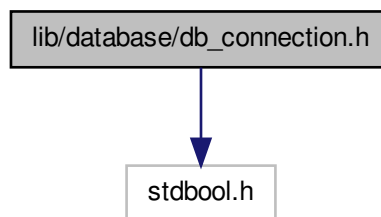
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 4.2 lib/database/db\_connection.h File Reference

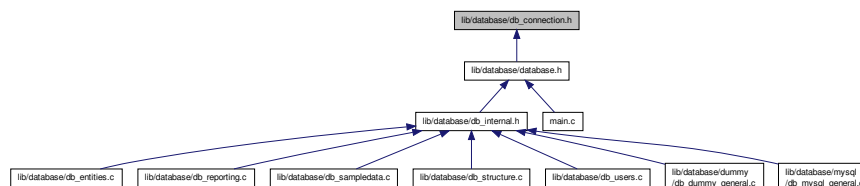
Interface to database connection functionality.

```
#include <stdbool.h>
```

Include dependency graph for db\_connection.h:



This graph shows which files directly or indirectly include this file:



## Functions

- bool **db\_connect** (const char \*host, const char \*database, const char \*username, const char \*password)  
*Connects to a database.*
- void **db\_close** (void)  
*Disconnects from a database.*



### 4.2.1 Detailed Description

Interface to database connection functionality. Function implementations are provided by the individual database components.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.2.2 Function Documentation

#### 4.2.2.1 `bool db_connect ( const char * host, const char * database, const char * username, const char * password )`

Connects to a database.

#### Parameters

<i>host</i>	The hostname.
<i>database</i>	The database name.
<i>username</i>	The username with which to connect.
<i>password</i>	The password for the specified user.

#### Returns

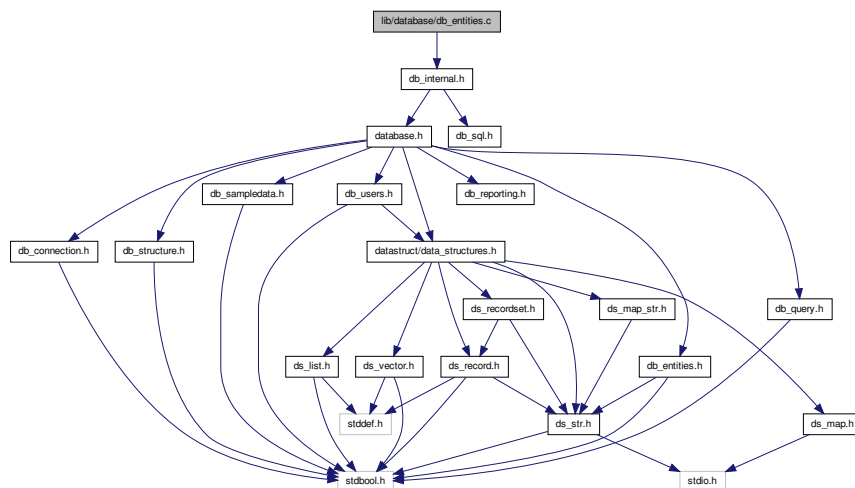
`true` if the connection was successfully made, `false` otherwise.

## 4.3 lib/database/db\_entities.c File Reference

Implementation of entities functionality.

```
#include "db_internal.h"
```

Include dependency graph for db\_entities.c:



## Functions

- bool `db_create_entities_table` (void)  
*Creates the entities table in the database.*
- bool `db_drop_entities_table` (void)  
*Drops the entities table in the database.*
- `ds_str` `db_list_entities_report` (void)  
*Creates a report listing all entities.*

### 4.3.1 Detailed Description

Implementation of entities functionality.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.3.2 Function Documentation

#### 4.3.2.1 bool `db_create_entities_table` ( void )

Creates the entities table in the database.

#### Returns

`true` on success, `false` on failure.

#### 4.3.2.2 bool `db_drop_entities_table` ( void )

Drops the entities table in the database.

#### Returns

`true` on success, `false` on failure.

#### 4.3.2.3 `ds_str` `db_list_entities_report` ( void )

Creates a report listing all entities.

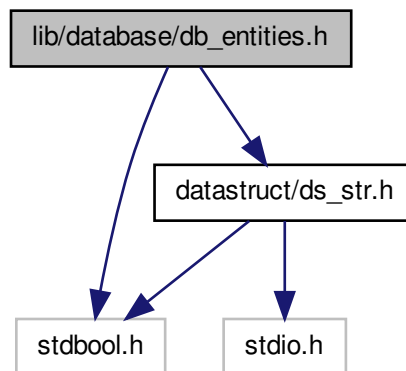
#### Returns

A `ds_str` containing the report.

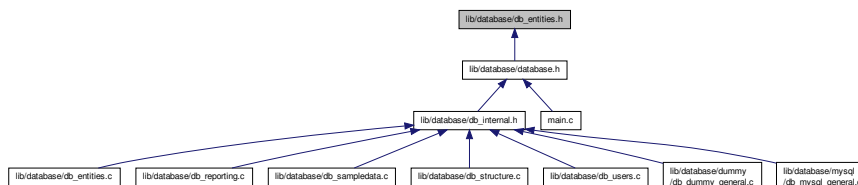
## 4.4 lib/database/db\_entities.h File Reference

Interface to entities functionality.

```
#include <stdbool.h>
#include "datastruct/ds_str.h"
Include dependency graph for db_entities.h:
```



This graph shows which files directly or indirectly include this file:



### Functions

- bool `db_create_entities_table` (void)  
*Creates the entities table in the database.*
- bool `db_drop_entities_table` (void)  
*Drops the entities table in the database.*
- `ds_str db_list_entities_report` (void)  
*Creates a report listing all entities.*

#### 4.4.1 Detailed Description

Interface to entities functionality.

**Author**

Paul Griffiths

**Copyright**

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

**4.4.2 Function Documentation****4.4.2.1 `bool db_create_entities_table ( void )`**

Creates the entities table in the database.

**Returns**

`true` on success, `false` on failure.

**4.4.2.2 `bool db_drop_entities_table ( void )`**

Drops the entities table in the database.

**Returns**

`true` on success, `false` on failure.

**4.4.2.3 `ds_str db_list_entities_report ( void )`**

Creates a report listing all entities.

## Returns

A `ds_str` containing the report.

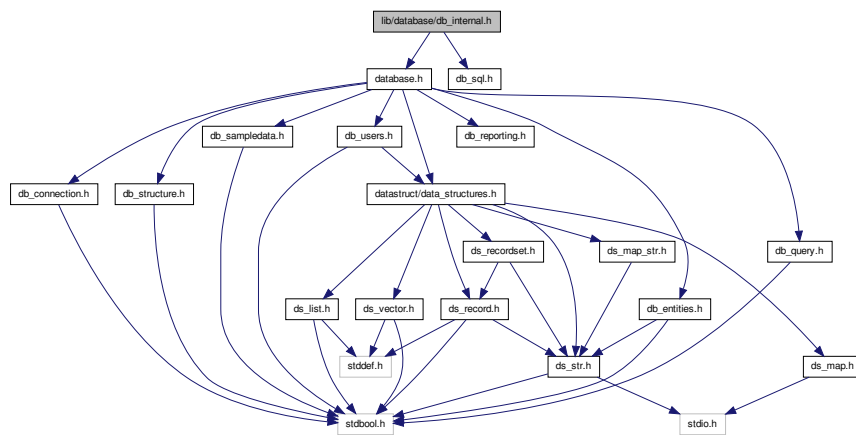
## 4.5 lib/database/db\_internal.h File Reference

Internal library interface to database functionality.

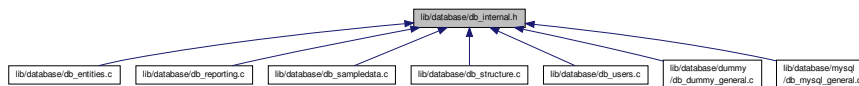
```
#include "database.h"
```

```
#include "db_sql.h"
```

Include dependency graph for `db_internal.h`:



This graph shows which files directly or indirectly include this file:



### 4.5.1 Detailed Description

Internal library interface to database functionality. The library interface includes the individual SQL functions which should be encapsulated from the user.

## Author

Paul Griffiths

## Copyright

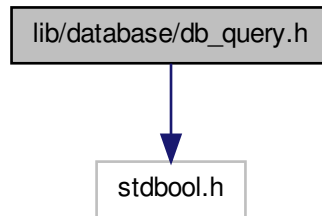
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 4.6 lib/database/db\_query.h File Reference

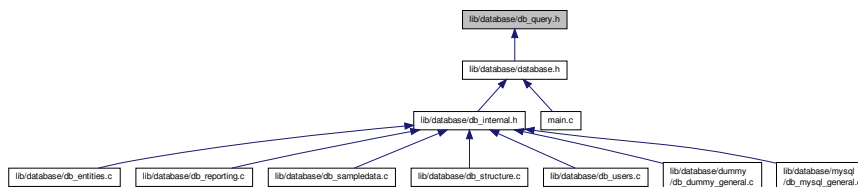
Interface to database query functionality.

```
#include <stdbool.h>
```

Include dependency graph for db\_query.h:



This graph shows which files directly or indirectly include this file:



## Functions

- bool [db\\_execute\\_query](#) (const char \*query)  
*Executes an SQL query on the database.*

### 4.6.1 Detailed Description

Interface to database query functionality. Function implementations are provided by the individual database components.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.6.2 Function Documentation

#### 4.6.2.1 bool db\_execute\_query ( const char \* query )

Executes an SQL query on the database.

## Parameters

<i>query</i>	The query to execute.
--------------	-----------------------

## Returns

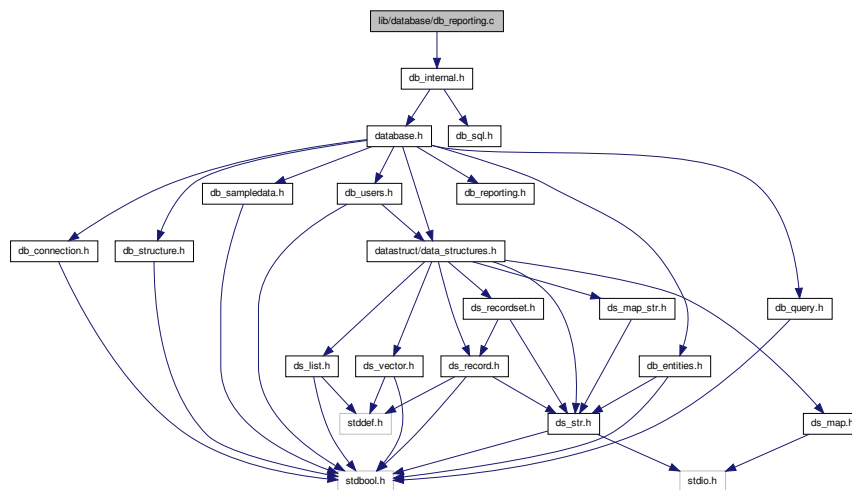
`true` if the query was successfully executed, `false` otherwise.

## 4.7 lib/database/db\_reporting.c File Reference

Implementation of database reporting functionality.

```
#include "db_internal.h"
```

Include dependency graph for db\_reporting.c:



## Functions

- [ds\\_str db\\_create\\_report\\_from\\_query](#) (const char \*query)

*Creates a text report from a query.*

## 4.7.1 Detailed Description

Implementation of database reporting functionality.

## Author

Paul Griffiths

## Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 4.7.2 Function Documentation

### 4.7.2.1 `ds_str db_create_report_from_query ( const char * query )`

Creates a text report from a query.

#### Parameters

<code>query</code>	The SELECT query to run.
--------------------	--------------------------

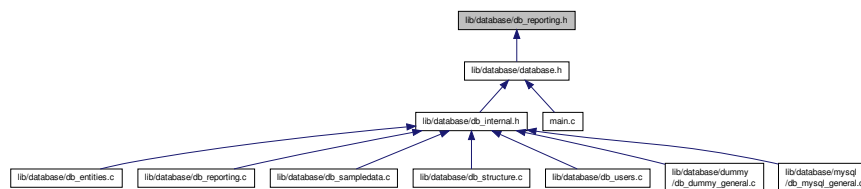
#### Returns

A `ds_str` containing the report, or `NULL` on failure.

## 4.8 lib/database/db\_reporting.h File Reference

Interface to database reporting functionality.

This graph shows which files directly or indirectly include this file:



## Functions

- `ds_str db_create_report_from_query (const char *query)`  
Creates a text report from a query.
- `ds_recordset db_create_recordset_from_query (const char *query)`  
Creates a `ds_recordset` from a query.

### 4.8.1 Detailed Description

Interface to database reporting functionality. Function implementations may be provided by the individual database components.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 4.8.2 Function Documentation

### 4.8.2.1 `ds_recordset db_create_recordset_from_query ( const char * query )`

Creates a `ds_recordset` from a query.



## Parameters

<i>query</i>	The SELECT query to run.
--------------	--------------------------

## Returns

A [ds\\_recordset](#) containing the query result, or `NULL` on failure.

4.8.2.2 `ds_str db_create_report_from_query ( const char * query )`

Creates a text report from a query.

## Parameters

<i>query</i>	The SELECT query to run.
--------------	--------------------------

## Returns

A [ds\\_str](#) containing the report, or `NULL` on failure.

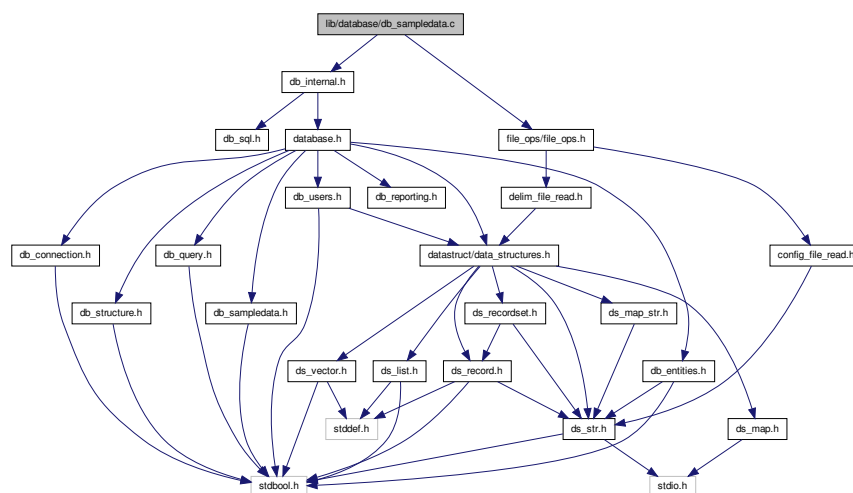
## 4.9 lib/database/db\_sampledata.c File Reference

Implementation of database sample data functionality.

```
#include "db_internal.h"
```

```
#include "file_ops/file_ops.h"
```

Include dependency graph for db\_sampledata.c:



## Functions

- bool [db\\_load\\_sample\\_data](#) (void)  
Loads sample data into the database.

## 4.9.1 Detailed Description

Implementation of database sample data functionality.

**Author**

Paul Griffiths

**Copyright**

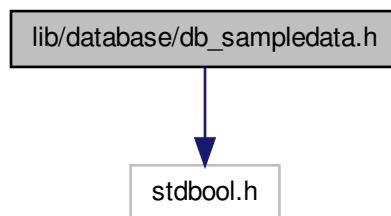
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

**4.10 lib/database/db\_sampledata.h File Reference**

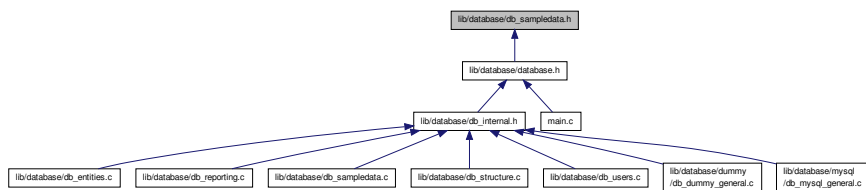
Interface to database sample data functionality.

```
#include <stdbool.h>
```

Include dependency graph for db\_sampledata.h:



This graph shows which files directly or indirectly include this file:

**Functions**

- bool [db\\_load\\_sample\\_data](#) (void)  
*Loads sample data into the database.*

**4.10.1 Detailed Description**

Interface to database sample data functionality.

**Author**

Paul Griffiths

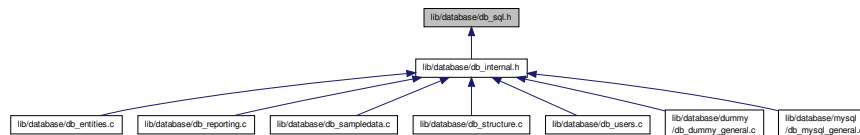
## Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 4.11 lib/database/db\_sql.h File Reference

Interface to database specific SQL strings.

This graph shows which files directly or indirectly include this file:



## Functions

- const char \* [db\\_create\\_users\\_table\\_sql](#) (void)
- const char \* [db\\_drop\\_users\\_table\\_sql](#) (void)
- const char \* [db\\_list\\_users\\_report\\_sql](#) (void)
- const char \* [db\\_create\\_entities\\_table\\_sql](#) (void)
- const char \* [db\\_drop\\_entities\\_table\\_sql](#) (void)
- const char \* [db\\_list\\_entities\\_report\\_sql](#) (void)

### 4.11.1 Detailed Description

Interface to database specific SQL strings. Function implementations are provided by the individual database components.

## Author

Paul Griffiths

## Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.11.2 Function Documentation

#### 4.11.2.1 const char\* db\_create\_entities\_table\_sql ( void )

**brief** Returns the SQL query to create the entities table.

## Returns

The SQL query.

#### 4.11.2.2 `const char* db_create_users_table_sql ( void )`

brief Returns the SQL query to create the users table.

##### Returns

The SQL query.

#### 4.11.2.3 `const char* db_drop_entities_table_sql ( void )`

brief Returns the SQL query to drop the entities table.

##### Returns

The SQL query.

#### 4.11.2.4 `const char* db_drop_users_table_sql ( void )`

brief Returns the SQL query to drop the users table.

##### Returns

The SQL query.

#### 4.11.2.5 `const char* db_list_entities_report_sql ( void )`

brief Returns the SQL query to run the "list entities" report.

##### Returns

The SQL query.

#### 4.11.2.6 `const char* db_list_users_report_sql ( void )`

brief Returns the SQL query to run the "list users" report.

##### Returns

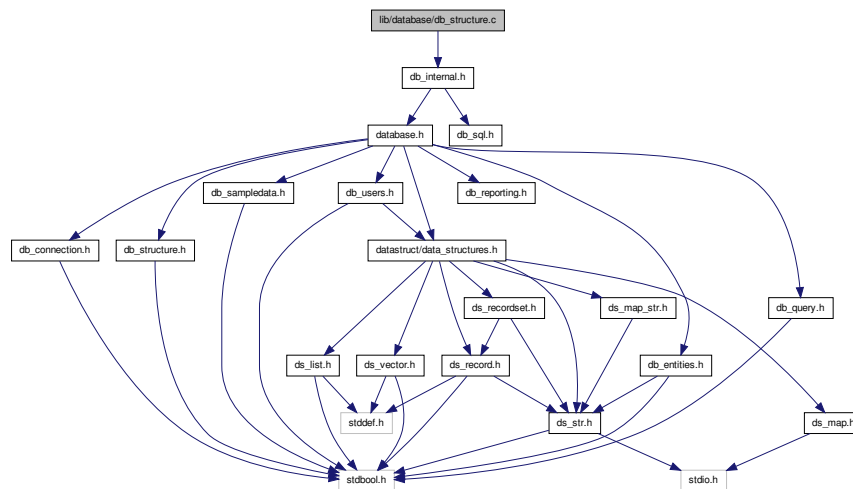
The SQL query.

## 4.12 `lib/database/db_structure.c` File Reference

Implementation of database structure functionality.

```
#include "db_internal.h"
```

Include dependency graph for db\_structure.c:



## Functions

- bool [db\\_create\\_database\\_structure](#) (void)  
*Creates an empty database structure.*
- bool [db\\_delete\\_database\\_structure](#) (void)  
*Deletes the database structure.*

### 4.12.1 Detailed Description

Implementation of database structure functionality.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.12.2 Function Documentation

#### 4.12.2.1 bool db\_create\_database\_structure ( void )

Creates an empty database structure.

#### Returns

true on success, false on failure.

#### 4.12.2.2 `bool db_delete_database_structure ( void )`

Deletes the database structure.

##### Returns

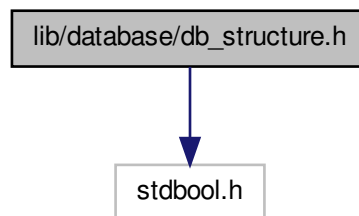
`true` on success, `false` on failure.

### 4.13 `lib/database/db_structure.h` File Reference

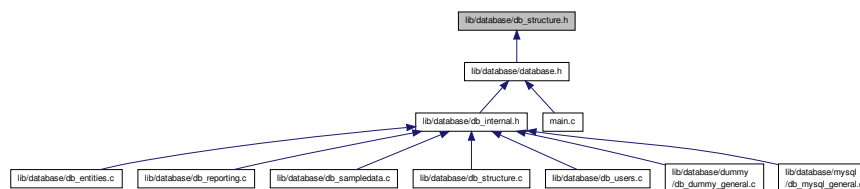
Interface to database structure functionality.

```
#include <stdbool.h>
```

Include dependency graph for `db_structure.h`:



This graph shows which files directly or indirectly include this file:



## Functions

- `bool db_create_database_structure (void)`  
*Creates an empty database structure.*
- `bool db_delete_database_structure (void)`  
*Deletes the database structure.*

#### 4.13.1 Detailed Description

Interface to database structure functionality.

**Author**

Paul Griffiths

**Copyright**

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

**4.13.2 Function Documentation****4.13.2.1 bool db\_create\_database\_structure ( void )**

Creates an empty database structure.

**Returns**

true on success, false on failure.

**4.13.2.2 bool db\_delete\_database\_structure ( void )**

Deletes the database structure.

**Returns**

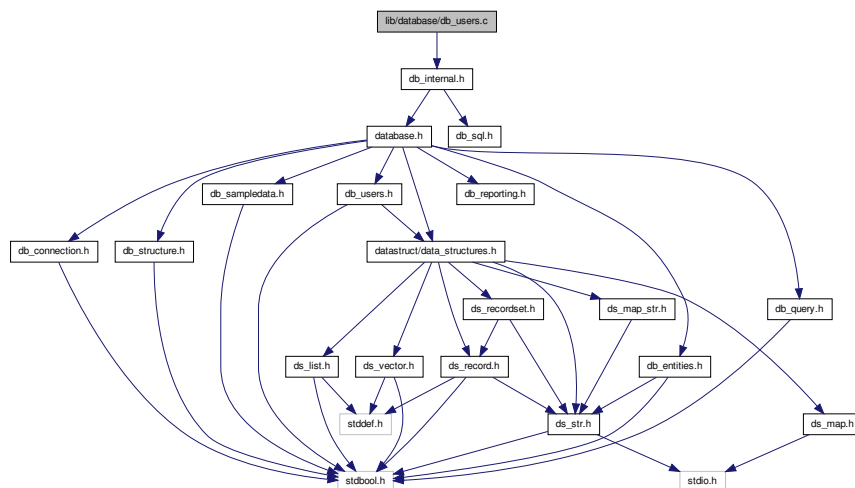
true on success, false on failure.

**4.14 lib/database/db\_users.c File Reference**

Implementation of users functionality.

```
#include "db_internal.h"
```

Include dependency graph for db\_users.c:



## Functions

- `bool db_create_users_table` (void)  
*Creates the users table in the database.*
- `bool db_drop_users_table` (void)  
*Drops the users table from the database.*
- `ds_str db_list_users_report` (void)  
*Creates a report listing all users.*

### 4.14.1 Detailed Description

Implementation of users functionality.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.14.2 Function Documentation

#### 4.14.2.1 `bool db_create_users_table ( void )`

Creates the users table in the database.

#### Returns

`true` on success, `false` on failure.

#### 4.14.2.2 `bool db_drop_users_table ( void )`

Drops the users table from the database.

#### Returns

`true` on success, `false` on failure.

#### 4.14.2.3 `ds_str db_list_users_report ( void )`

Creates a report listing all users.

#### Returns

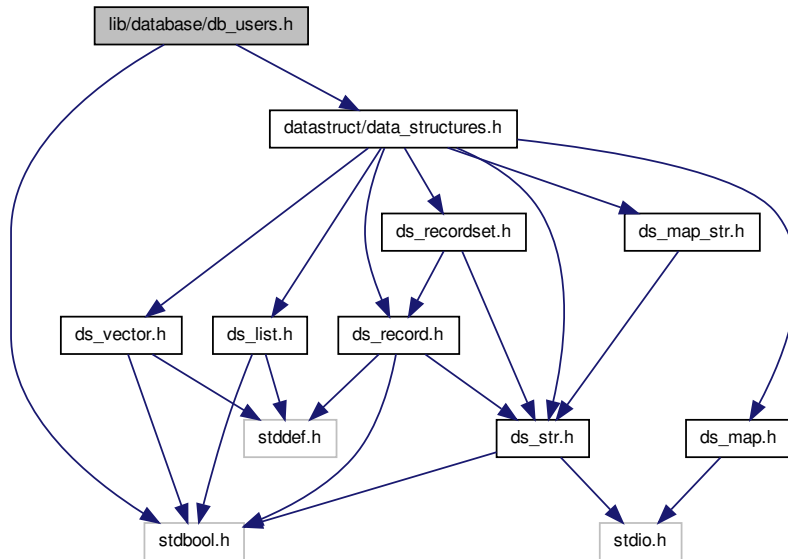
A `ds_str` containing the report.



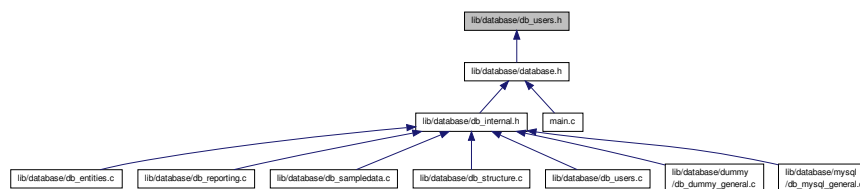
## 4.15 lib/database/db\_users.h File Reference

Interface to users functionality.

```
#include <stdbool.h>
#include "datastruct/data_structures.h"
Include dependency graph for db_users.h:
```



This graph shows which files directly or indirectly include this file:



### Functions

- bool [db\\_create\\_users\\_table](#) (void)  
*Creates the users table in the database.*
- bool [db\\_drop\\_users\\_table](#) (void)  
*Drops the users table from the database.*
- [ds\\_str db\\_list\\_users\\_report](#) (void)  
*Creates a report listing all users.*

#### 4.15.1 Detailed Description

Interface to users functionality.

**Author**

Paul Griffiths

**Copyright**

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

**4.15.2 Function Documentation****4.15.2.1 `bool db_create_users_table ( void )`**

Creates the users table in the database.

**Returns**

`true` on success, `false` on failure.

**4.15.2.2 `bool db_drop_users_table ( void )`**

Drops the users table from the database.

**Returns**

`true` on success, `false` on failure.

**4.15.2.3 `ds_str db_list_users_report ( void )`**

Creates a report listing all users.

**Returns**

A `ds_str` containing the report.

**4.16 `lib/database/dummy/db_dummy_create_entities_table_sql.c` File Reference**

Returns dummy SQL query to create entities table.

**Functions**

- `const char * db_create_entities_table_sql (void)`

**4.16.1 Detailed Description**

Returns dummy SQL query to create entities table.

**Author**

Paul Griffiths

**Copyright**

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 4.16.2 Function Documentation

### 4.16.2.1 `const char* db.create_entities_table_sql ( void )`

brief Returns the SQL query to create the entities table.

Returns

The SQL query.

## 4.17 lib/database/dummy/db\_dummy\_create\_users\_table\_sql.c File Reference

Returns dummy SQL query to create users table.

### Functions

- `const char * db_create_users_table_sql (void)`

### 4.17.1 Detailed Description

Returns dummy SQL query to create users table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 4.17.2 Function Documentation

### 4.17.2.1 `const char* db.create_users_table_sql ( void )`

brief Returns the SQL query to create the users table.

Returns

The SQL query.

## 4.18 lib/database/dummy/db\_dummy\_drop\_entities\_table\_sql.c File Reference

Returns dummy SQL query to drop entities table.

### Functions

- `const char * db_drop_entities_table_sql (void)`

### 4.18.1 Detailed Description

Returns dummy SQL query to drop entities table.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.18.2 Function Documentation

#### 4.18.2.1 `const char* db_drop_entities_table_sql ( void )`

brief Returns the SQL query to drop the entities table.

#### Returns

The SQL query.

## 4.19 lib/database/dummy/db\_dummy\_drop\_users\_table\_sql.c File Reference

Returns dummy SQL query to drop users table.

### Functions

- `const char * db_drop_users_table_sql (void)`

### 4.19.1 Detailed Description

Returns dummy SQL query to drop users table.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.19.2 Function Documentation

#### 4.19.2.1 `const char* db_drop_users_table_sql ( void )`

brief Returns the SQL query to drop the users table.

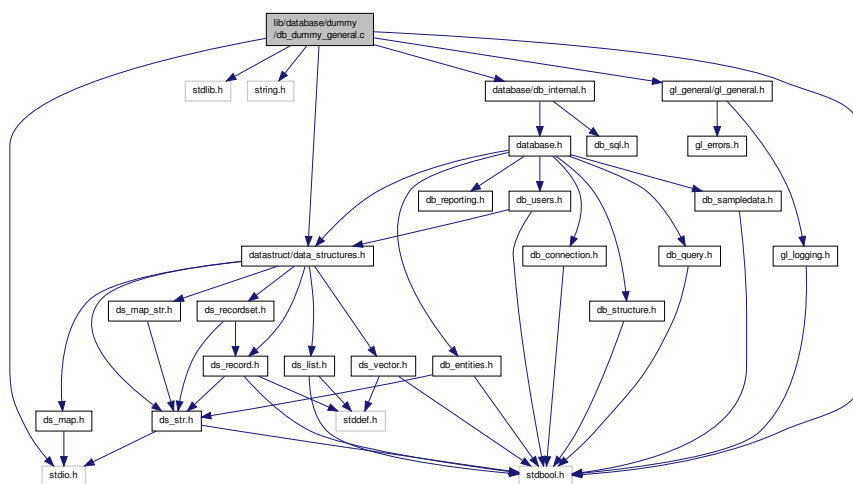
#### Returns

The SQL query.

## 4.20 lib/database/dummy/db\_dummy\_general.c File Reference

Implementation of dummy database functionality.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include "gl_general/gl_general.h"
#include "database/db_internal.h"
#include "datastruct/data_structures.h"
Include dependency graph for db_dummy_general.c:
```



### Macros

- `#define _XOPEN_SOURCE 600`

### Functions

- `bool db_connect (const char *host, const char *database, const char *username, const char *password)`  
*Connects to a database.*
- `void db_close (void)`  
*Disconnects from a database.*
- `bool db_execute_query (const char *query)`  
*Executes an SQL query on the database.*
- `ds_recordset db_create_recordset_from_query (const char *query)`  
*Creates a `ds_recordset` from a query.*

#### 4.20.1 Detailed Description

Implementation of dummy database functionality. This module is useful when compiling for testing purpose on a system without any of the supported database development libraries available.

#### Author

Paul Griffiths

## Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 4.20.2 Function Documentation

### 4.20.2.1 `bool db_connect ( const char * host, const char * database, const char * username, const char * password )`

Connects to a database.

#### Parameters

<i>host</i>	The hostname.
<i>database</i>	The database name.
<i>username</i>	The username with which to connect.
<i>password</i>	The password for the specified user.

#### Returns

`true` if the connection was successfully made, `false` otherwise.

### 4.20.2.2 `ds_recordset db_create_recordset_from_query ( const char * query )`

Creates a `ds_recordset` from a query.

#### Parameters

<i>query</i>	The SELECT query to run.
--------------	--------------------------

#### Returns

A `ds_recordset` containing the query result, or `NULL` on failure.

### 4.20.2.3 `bool db_execute_query ( const char * query )`

Executes an SQL query on the database.

#### Parameters

<i>query</i>	The query to execute.
--------------	-----------------------

#### Returns

`true` if the query was successfully executed, `false` otherwise.

## 4.21 `lib/database/dummy/db_dummy_list_entities_report_sql.c` File Reference

Returns dummy SQL query to create list entities report.

## Functions

- `const char * db\_list\_entities\_report\_sql (void)`

### 4.21.1 Detailed Description

Returns dummy SQL query to create list entities report.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.21.2 Function Documentation

#### 4.21.2.1 `const char* db_list_entities_report_sql ( void )`

brief Returns the SQL query to run the "list entities" report.

#### Returns

The SQL query.

## 4.22 lib/database/dummy/db\_dummy\_list\_users\_report\_sql.c File Reference

Returns dummy SQL query to create list users report.

### Functions

- `const char * db_list_users_report_sql (void)`

### 4.22.1 Detailed Description

Returns dummy SQL query to create list users report.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.22.2 Function Documentation

#### 4.22.2.1 `const char* db_list_users_report_sql ( void )`

brief Returns the SQL query to run the "list users" report.

#### Returns

The SQL query.

## 4.23 lib/database/mysql/db\_mysql\_create\_entities\_table\_sql.c File Reference

Returns MYSQL SQL query to create entities table.

### Functions

- const char \* [db\\_create\\_entities\\_table\\_sql](#) (void)

#### 4.23.1 Detailed Description

Returns MYSQL SQL query to create entities table.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

#### 4.23.2 Function Documentation

##### 4.23.2.1 const char\* db\_create\_entities\_table\_sql ( void )

brief Returns the SQL query to create the entities table.

#### Returns

The SQL query.

## 4.24 lib/database/mysql/db\_mysql\_create\_users\_table\_sql.c File Reference

Returns MYSQL SQL query to create users table.

### Functions

- const char \* [db\\_create\\_users\\_table\\_sql](#) (void)

#### 4.24.1 Detailed Description

Returns MYSQL SQL query to create users table.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>



## 4.24.2 Function Documentation

### 4.24.2.1 `const char* db.create_users_table_sql ( void )`

brief Returns the SQL query to create the users table.

Returns

The SQL query.

## 4.25 lib/database/mysql/db\_mysql\_drop\_entities\_table\_sql.c File Reference

Returns MYSQL SQL query to drop entities table.

### Functions

- `const char * db_drop_entities_table_sql (void)`

### 4.25.1 Detailed Description

Returns MYSQL SQL query to drop entities table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 4.25.2 Function Documentation

### 4.25.2.1 `const char* db_drop_entities_table_sql ( void )`

brief Returns the SQL query to drop the entities table.

Returns

The SQL query.

## 4.26 lib/database/mysql/db\_mysql\_drop\_users\_table\_sql.c File Reference

Returns MYSQL SQL query to drop users table.

### Functions

- `const char * db_drop_users_table_sql (void)`

### 4.26.1 Detailed Description

Returns MYSQL SQL query to drop users table.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.26.2 Function Documentation

#### 4.26.2.1 `const char* db_drop_users_table_sql ( void )`

brief Returns the SQL query to drop the users table.

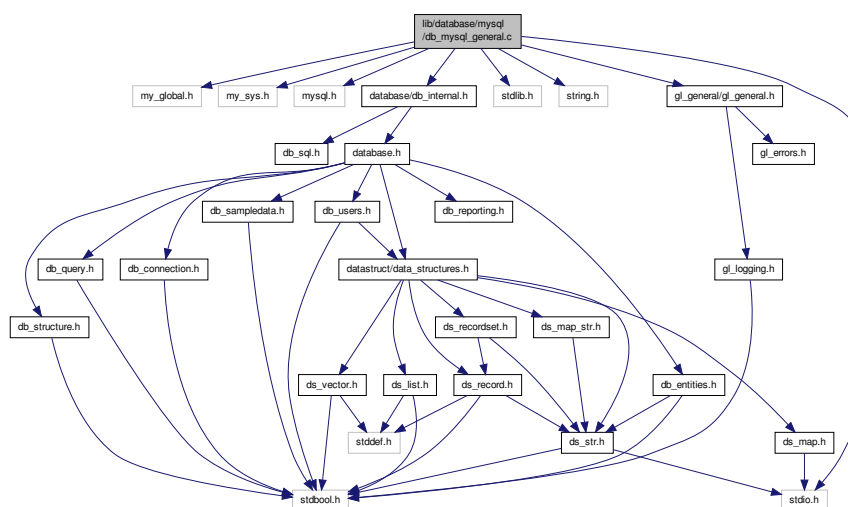
#### Returns

The SQL query.

## 4.27 lib/database/mysql/db\_mysql\_general.c File Reference

Implementation of MYSQL database functionality.

```
#include <my_global.h>
#include <my_sys.h>
#include <mysql.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "gl_general/gl_general.h"
#include "database/db_internal.h"
Include dependency graph for db_mysql_general.c:
```



## Functions

- bool [db\\_connect](#) (const char \*host, const char \*database, const char \*username, const char \*password)  
*Connects to a database.*
- void [db\\_close](#) (void)  
*Disconnects from a database.*
- bool [db\\_execute\\_query](#) (const char \*query)  
*Executes an SQL query on the database.*
- [ds\\_recordset](#) [db\\_create\\_recordset\\_from\\_query](#) (const char \*query)  
*Creates a [ds\\_recordset](#) from a query.*

## Variables

- MYSQL \* [main\\_mss](#) = NULL
- MYSQL \* [conn\\_mss](#) = NULL

### 4.27.1 Detailed Description

Implementation of MYSQL database functionality.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.27.2 Function Documentation

#### 4.27.2.1 bool db\_connect ( const char \* host, const char \* database, const char \* username, const char \* password )

Connects to a database.

##### Parameters

<i>host</i>	The hostname.
<i>database</i>	The database name.
<i>username</i>	The username with which to connect.
<i>password</i>	The password for the specified user.

##### Returns

`true` if the connection was successfully made, `false` otherwise.

#### 4.27.2.2 ds\_recordset db\_create\_recordset\_from\_query ( const char \* query )

Creates a [ds\\_recordset](#) from a query.

##### Parameters

<i>query</i>	The SELECT query to run.
--------------	--------------------------

**Returns**

A `ds_recordset` containing the query result, or `NULL` on failure.

**4.27.2.3 bool db\_execute\_query ( const char \* *query* )**

Executes an SQL query on the database.

**Parameters**

<i>query</i>	The query to execute.
--------------	-----------------------

**Returns**

`true` if the query was successfully executed, `false` otherwise.

**4.27.3 Variable Documentation****4.27.3.1 MYSQL\* conn\_mss = NULL**

MYSQL connection object.

**4.27.3.2 MYSQL\* main\_mss = NULL**

MYSQL initialization object.

**4.28 lib/database/mysql/db\_mysql\_list\_entities\_report\_sql.c File Reference**

Returns MYSQL SQL query to create list entities report.

**Functions**

- const char \* `db_list_entities_report_sql` (void)

**4.28.1 Detailed Description**

Returns MYSQL SQL query to create list entities report.

**Author**

Paul Griffiths

**Copyright**

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

**4.28.2 Function Documentation****4.28.2.1 const char\* db\_list\_entities\_report\_sql ( void )**

brief Returns the SQL query to run the "list entities" report.

#### Returns

The SQL query.

## 4.29 lib/database/mysql/db\_mysql\_list\_users\_report\_sql.c File Reference

Returns MYSQL SQL query to create list users report.

#### Functions

- const char \* [db\\_list\\_users\\_report\\_sql](#) (void)

### 4.29.1 Detailed Description

Returns MYSQL SQL query to create list users report.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.29.2 Function Documentation

#### 4.29.2.1 const char\* db\_list\_users\_report\_sql ( void )

brief Returns the SQL query to run the "list users" report.

#### Returns

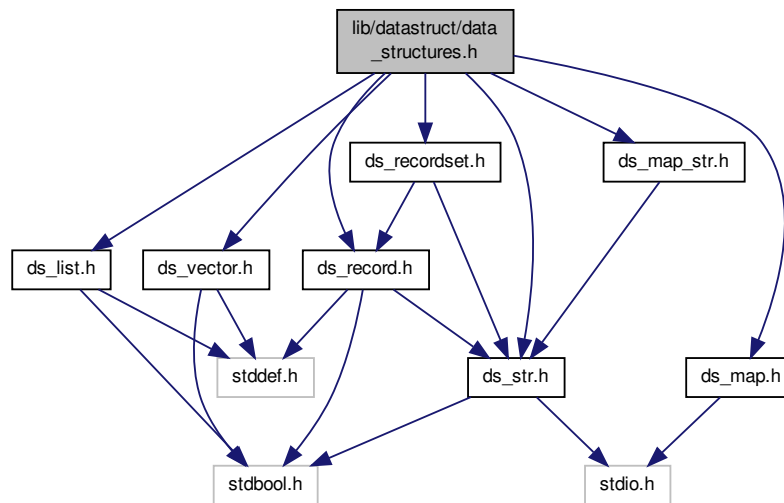
The SQL query.

## 4.30 lib/datastruct/data\_structures.h File Reference

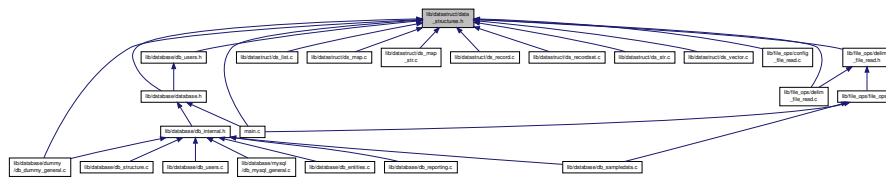
Interface to data structures.

```
#include "ds_list.h"
#include "ds_vector.h"
#include "ds_str.h"
#include "ds_map.h"
#include "ds_map_str.h"
#include "ds_record.h"
#include "ds_recordset.h"
```

Include dependency graph for data\_structures.h:



This graph shows which files directly or indirectly include this file:



### 4.30.1 Detailed Description

Interface to data structures.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 4.31 lib/datastruct/ds\_list.c File Reference

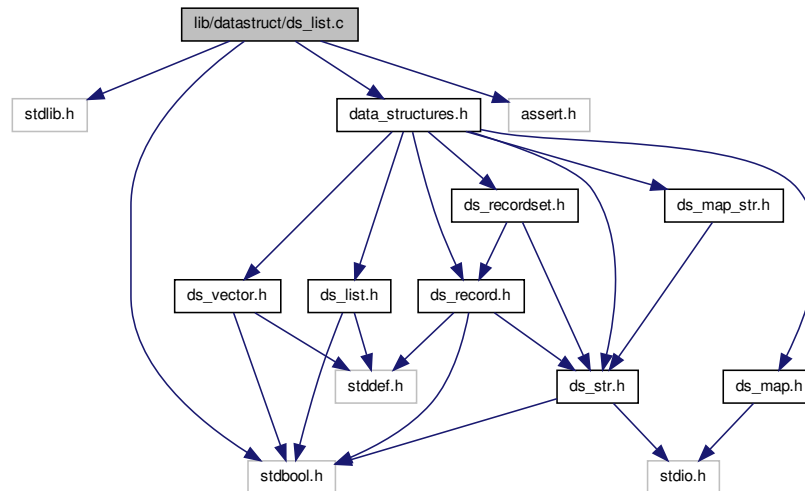
Implementation of generic doubly-linked list data structure.

```

#include <stdlib.h>
#include <stdbool.h>
#include <assert.h>
#include "data_structures.h"

```

Include dependency graph for ds\_list.c:



## Data Structures

- struct [ds\\_list\\_element](#)
- struct [ds\\_list](#)

## Functions

- struct [ds\\_list](#) \* [ds\\_list\\_create](#) (const bool free\_on\_delete, void(\*destructor)(void \*))  
*Creates a new list.*
- void [ds\\_list\\_destroy](#) (struct [ds\\_list](#) \*list)
- void [ds\\_list\\_destructor](#) (void \*list)  
*A list destructor function.*
- struct [ds\\_list](#) \* [ds\\_list\\_append](#) (struct [ds\\_list](#) \*list, void \*data)
- void [ds\\_list\\_remove\\_tail](#) (struct [ds\\_list](#) \*list)
- void [ds\\_list\\_remove\\_all](#) (struct [ds\\_list](#) \*list)
- void \* [ds\\_list\\_element](#) (struct [ds\\_list](#) \*list, const size\_t index)
- size\_t [ds\\_list\\_length](#) (struct [ds\\_list](#) \*list)
- bool [ds\\_list\\_is\\_empty](#) (struct [ds\\_list](#) \*list)
- void [ds\\_list\\_seek\\_start](#) (struct [ds\\_list](#) \*list)
- void [ds\\_list\\_seek\\_end](#) (struct [ds\\_list](#) \*list)
- void \* [ds\\_list\\_get\\_next\\_data](#) (struct [ds\\_list](#) \*list)
- void \* [ds\\_list\\_get\\_prev\\_data](#) (struct [ds\\_list](#) \*list)

### 4.31.1 Detailed Description

Implementation of generic doubly-linked list data structure.

Author

Paul Griffiths

## Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 4.31.2 Function Documentation

4.31.2.1 `struct ds_list* ds_list_create ( const bool free_on_delete, void(*)(void *) destructor )` [read]

Creates a new list.

### Parameters

<i>free_on_delete</i>	Set to <code>true</code> if the list elements should be destroyed when removed from the list, and when the list itself is destroyed. If set to <code>false</code> , the caller is responsible for destroying the elements prior to destroying the list.
<i>destructor</i>	Pointer to a destructor function to use for destroying the list elements, when <code>free_on_delete</code> is true. If this is set to <code>NULL</code> , <code>free()</code> from the standard C library will be used to destroy the elements.

### Returns

A newly created list, or `NULL` on failure.

4.31.2.2 `void ds_list_destructor ( void * list )`

A list destructor function.

This function may be passed to `ds_list_create()` when creating a list of lists. It calls `ds_list_destroy()`, but the parameter of `ds_list_destroy()` is not compatible with the function signature expected by `ds_list_create()`, so this function provides an appropriate interface.

### Parameters

<i>list</i>	The list to destroy.
-------------	----------------------

## 4.32 lib/datastruct/ds\_list.h File Reference

Interface to generic doubly-linked list data structure.

```
#include <stddef.h>
#include <stdbool.h>
```



$$f_{\text{max}} = \frac{1}{2\pi} \sqrt{\frac{1}{L \cdot C_{\text{eff}}}} \quad (1)$$



- typedef struct **ds\_list** \* **ds\_list**

- `ds_list_ds_list_create` (const bool free\_on\_delete, void(\*destructor)(void \*))  
*Creates a new list.*
- void `ds_list_destroy` (`ds_list` list)  
*Destroys a list and frees any associated resources.*
- void `ds_list_destructor` (void \*list)  
*A list destructor function.*
- `ds_list_ds_list_append` (`ds_list` list, void \*element)  
*Appends an element to a list.*
- void `ds_list_remove_tail` (`ds_list` list)  
*Removes the last element of a list.*
- void `ds_list_remove_all` (`ds_list` list)  
*Removes all the elements from a list.*
- void \* `ds_list_element` (`ds_list` list, const size\_t index)  
*Retrieves the data at a specified index.*
- size\_t `ds_list_length` (`ds_list` list)  
*Returns the number of elements in a list.*
- bool `ds_list_is_empty` (`ds_list` list)  
*Checks if a list is empty.*

- void `ds_list_seek_start` (`ds_list` list)  
*Sets the current element to the first element of a list.*
- void `ds_list_seek_end` (`ds_list` list)  
*Sets the current element to the last element of a list.*
- void \* `ds_list_get_next_data` (`ds_list` list)  
*Returns the next element of the list.*
- void \* `ds_list_get_prev_data` (`ds_list` list)  
*Returns the previous element of the list.*

### 4.32.1 Detailed Description

Interface to generic doubly-linked list data structure.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.32.2 Typedef Documentation

#### 4.32.2.1 typedef struct `ds_list`\* `ds_list`

Typedef for opaque list datatype

### 4.32.3 Function Documentation

#### 4.32.3.1 `ds_list` `ds_list.append` ( `ds_list` list, void \* element )

Appends an element to a list.

#### Parameters

<i>list</i>	The list to which to append.
<i>element</i>	The element to append.

#### Returns

The same list, or `NULL` on failure.

#### 4.32.3.2 `ds_list` `ds_list.create` ( const bool *free\_on\_delete*, void(\*) (void \*) *destructor* ) [read]

Creates a new list.

#### Parameters

<i>free_on_delete</i>	Set to <code>true</code> if the list elements should be destroyed when removed from the list, and when the list itself is destroyed. If set to <code>false</code> , the caller is responsible for destroying the elements prior to destroying the list.
<i>destructor</i>	Pointer to a destructor function to use for destroying the list elements, when <i>free_on_delete</i> is true. If this is set to <code>NULL</code> , <code>free()</code> from the standard C library will be used to destroy the elements.

**Returns**

A newly created list, or `NULL` on failure.

**4.32.3.3 void ds\_list\_destroy ( ds\_list list )**

Destroys a list and frees any associated resources.

**Parameters**

<i>list</i>	The list to destroy.
-------------	----------------------

**4.32.3.4 void ds\_list\_destructor ( void \* list )**

A list destructor function.

This function may be passed to `ds_list_create()` when creating a list of lists. It calls `ds_list_destroy()`, but the parameter of `ds_list_destroy()` is not compatible with the function signature expected by `ds_list_create()`, so this function provides an appropriate interface.

**Parameters**

<i>list</i>	The list to destroy.
-------------	----------------------

**4.32.3.5 void\* ds\_list\_element ( ds\_list list, const size\_t index )**

Retrieves the data at a specified index.

**Parameters**

<i>list</i>	The list from which to retrieve.
<i>index</i>	The index of the desired element.

**Returns**

A pointer to the data, or `NULL` if the index is out of range.

**4.32.3.6 void\* ds\_list\_get\_next\_data ( ds\_list list )**

Returns the next element of the list.

This function returns the data of the "current element", and advances the current element pointer. Subsequent calls to this function will return successive elements.

**Parameters**

<i>list</i>	The list.
-------------	-----------

**Returns**

A pointer to the next element, or `NULL` if the end of the list has been reached.

**4.32.3.7 void\* ds\_list\_get\_prev.data ( ds\_list list )**

Returns the previous element of the list.

This function returns the data of the "current element", and decrements the current element pointer. Subsequent calls to this function will return successively earlier elements.

**Parameters**

<i>list</i>	The list.
-------------	-----------

**Returns**

A pointer to the previous element, or `NULL` if the start of the list has been reached.

**4.32.3.8 bool ds\_list\_is\_empty ( ds\_list list )**

Checks if a list is empty.

**Parameters**

<i>list</i>	The list to check.
-------------	--------------------

**Returns**

`true` if the list is empty, `false` otherwise.

**4.32.3.9 size\_t ds\_list\_length ( ds\_list list )**

Returns the number of elements in a list.

**Parameters**

<i>list</i>	The list.
-------------	-----------

**Returns**

The number of elements in the list.

**4.32.3.10 void ds\_list\_remove\_all ( ds\_list list )**

Removes all the elements from a list.

**Parameters**

<i>list</i>	The list from which to remove.
-------------	--------------------------------

4.32.3.11 void ds\_list\_remove\_tail ( ds\_list list )

Removes the last element of a list.

Parameters

<i>list</i>	The list from which to remove.
-------------	--------------------------------

4.32.3.12 void ds\_list\_seek\_end ( ds\_list list )

Sets the current element to the last element of a list.

Parameters

<i>list</i>	The list.
-------------	-----------

4.32.3.13 void ds\_list\_seek\_start ( ds\_list list )

Sets the current element to the first element of a list.

Parameters

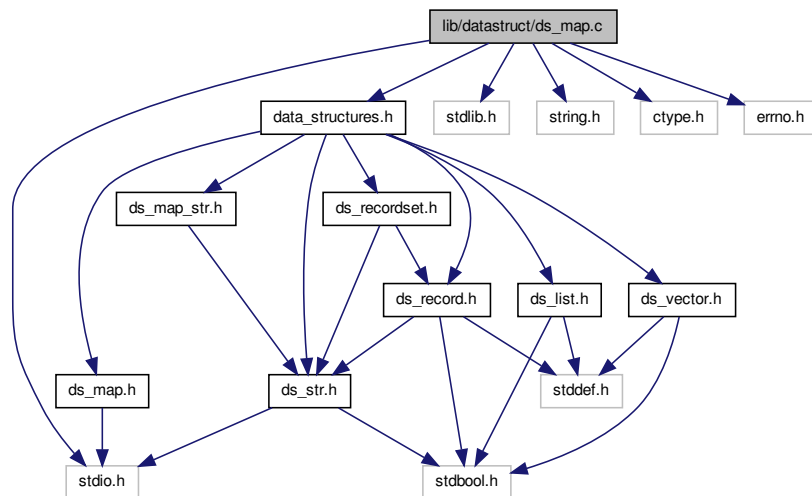
<i>list</i>	The list.
-------------	-----------

## 4.33 lib/datastruct/ds\_map.c File Reference

Implementation of string-string hash map data structure.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <errno.h>
#include "data_structures.h"
```

Include dependency graph for ds\_map.c:



## Data Structures

- struct [kv\\_pair\\_node](#)
- struct [ds\\_map](#)

## Macros

- `#define \_POSIX\_C\_SOURCE 200809L`  
*Enables POSIX library functions.*

## Functions

- struct [ds\\_map](#) \* [ds\\_map\\_init](#) (const size\_t hash\_size)  
*Initializes a hash map.*
- void [ds\\_map\\_destroy](#) (struct [ds\\_map](#) \*map)
- const char \* [ds\\_map\\_get\\_value](#) (struct [ds\\_map](#) \*map, const char \*key)
- void [ds\\_map\\_insert](#) (struct [ds\\_map](#) \*map, const char \*key, const char \*value)
- void [ds\\_map\\_print\\_all](#) (ds\_map map, FILE \*outfile)  
*Prints all the key-value pairs in a map to stdout.*

### 4.33.1 Detailed Description

Implementation of string-string hash map data structure.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

#### 4.33.2.1 struct ds\_map\* ds\_map\_init ( const size\_t hash\_size ) [read]

## Parameters

<i>hash_size</i>	The number of possible hash values.
------------------	-------------------------------------

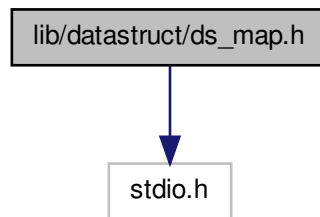
A reference to the newly-created hash map.

Prints all the key-value pairs in a map to stdout.

<i>map</i>	A reference to the map.
<i>outfile</i>	A FILE pointer to which to print the output.

Interface to string-string hash map data structure.

Include dependency graph for ds\_map.h:

[illegible]

## Typedefs

- typedef struct [ds\\_map](#) \* [ds\\_map](#)

## Functions

- [ds\\_map ds\\_map\\_init](#) (const size\_t hash\_size)  
*Initializes a hash map.*
- void [ds\\_map\\_destroy](#) ([ds\\_map](#) map)  
*Destroys a hash map.*
- const char \* [ds\\_map\\_get\\_value](#) ([ds\\_map](#) map, const char \*key)  
*Retrieves a value associated with a key in the map.*
- void [ds\\_map\\_insert](#) ([ds\\_map](#) map, const char \*key, const char \*value)  
*Inserts a key-value pair into a map.*
- void [ds\\_map\\_print\\_all](#) ([ds\\_map](#) map, FILE \*outfile)  
*Prints all the key-value pairs in a map to stdout.*

### 4.34.1 Detailed Description

Interface to string-string hash map data structure.

#### Author

Paul Griffiths

#### Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.34.2 Typedef Documentation

#### 4.34.2.1 typedef struct [ds\\_map](#)\* [ds\\_map](#)

Opaque data type for hash map

### 4.34.3 Function Documentation

#### 4.34.3.1 void [ds\\_map\\_destroy](#) ( [ds\\_map](#) map )

Destroys a hash map.

#### Parameters

<i>map</i>	A reference to the map to destroy.
------------	------------------------------------

#### 4.34.3.2 const char\* [ds\\_map\\_get\\_value](#) ( [ds\\_map](#) map, const char \* key )

Retrieves a value associated with a key in the map.



## Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.

## Returns

A pointer to the value associated with the key, or `NULL` if the key is not in the map. The caller should not modify the string to which this pointer points.

4.34.3.3 `ds_map ds_map_init ( const size_t hash_size )` [read]

Initializes a hash map.

## Parameters

<i>hash_size</i>	The number of possible hash values.
------------------	-------------------------------------

## Returns

A reference to the newly-created hash map.

4.34.3.4 `void ds_map_insert ( ds_map map, const char * key, const char * value )`

Inserts a key-value pair into a map.

The key and value are copied, so the caller may modify or `free()` them after calling this function.

## Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.
<i>value</i>	The value.

4.34.3.5 `void ds_map_print_all ( ds_map map, FILE * outfile )`

Prints all the key-value pairs in a map to stdout.

## Parameters

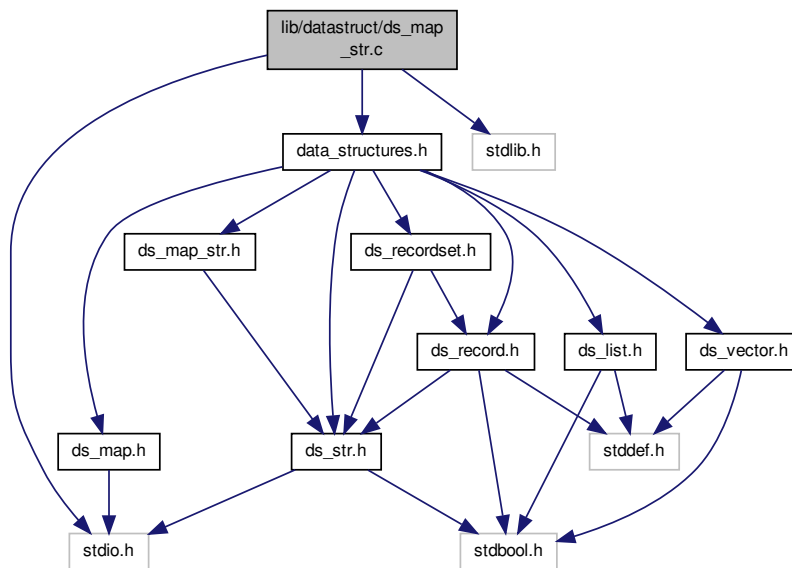
<i>map</i>	A reference to the map.
<i>outfile</i>	A FILE pointer to which to print the output.

## 4.35 lib/datastruct/ds\_map\_str.c File Reference

Implementation of string-string hash map data structure.

```
#include <stdio.h>
#include <stdlib.h>
#include "data_structures.h"
```

Include dependency graph for `ds_map_str.c`:



## Data Structures

- struct `kv_pair_node`
- struct `ds_map_str`

## Functions

- struct `ds_map_str` \* `ds_map_str_init` (const size\_t hash\_size)  
*Initializes a hash map.*
- void `ds_map_str_destroy` (struct `ds_map_str` \*map)
- `ds_str` `ds_map_str_get_value` (struct `ds_map_str` \*map, `ds_str` key)
- void `ds_map_str_insert` (struct `ds_map_str` \*map, `ds_str` key, `ds_str` value)

### 4.35.1 Detailed Description

Implementation of string-string hash map data structure.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>



## Functions

- `ds_map_str ds_map_str_init` (const size\_t hash\_size)  
*Initializes a hash map.*
- void `ds_map_str_destroy` (ds\_map\_str map)  
*Destroys a hash map.*
- `ds_str ds_map_str_get_value` (ds\_map\_str map, ds\_str key)  
*Retrieves a value associated with a key in the map.*
- void `ds_map_str_insert` (ds\_map\_str map, ds\_str key, ds\_str value)  
*Inserts a key-value pair into a map.*

### 4.36.1 Detailed Description

Interface to string-string hash map data structure.

#### Author

Paul Griffiths

#### Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.36.2 Typedef Documentation

#### 4.36.2.1 typedef struct ds\_map\_str\* ds\_map\_str

Opaque data type for hash map

### 4.36.3 Function Documentation

#### 4.36.3.1 void ds\_map\_str\_destroy ( ds\_map\_str map )

Destroys a hash map.

#### Parameters

<i>map</i>	A reference to the map to destroy.
------------	------------------------------------

#### 4.36.3.2 ds\_str ds\_map\_str\_get\_value ( ds\_map\_str map, ds\_str key )

Retrieves a value associated with a key in the map.

#### Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.

#### Returns

A pointer to the value associated with the key, or `NULL` if the key is not in the map. The caller should not modify the string to which this pointer points.

4.36.3.3 `ds_map_str ds_map_str_init ( const size_t hash_size )` [read]

Initializes a hash map.

## Parameters

<i>hash_size</i>	The number of possible hash values.
------------------	-------------------------------------

## Returns

A reference to the newly-created hash map.

4.36.3.4 `void ds_map_str_insert ( ds_map_str map, ds_str key, ds_str value )`

Inserts a key-value pair into a map.

The key and value are copied, so the caller may modify or `free()` them after calling this function.

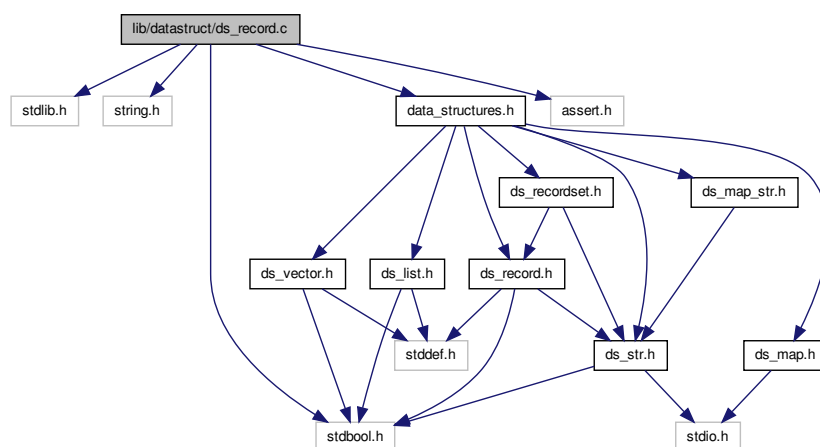
## Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.
<i>value</i>	The value.

## 4.37 lib/datastruct/ds\_record.c File Reference

Implementation of record database structure.

```
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <assert.h>
#include "data_structures.h"
Include dependency graph for ds_record.c:
```



## Data Structures

- struct [ds\\_record](#)

## Functions

- struct [ds\\_record](#) \* [ds\\_record\\_create](#) (const size\_t size)  
*Creates a new record.*
- void [ds\\_record\\_destroy](#) (struct [ds\\_record](#) \*record)
- void [ds\\_record\\_destructor](#) (void \*record)  
*A record destructor function.*
- void [ds\\_record\\_clear](#) (struct [ds\\_record](#) \*record)
- void [ds\\_record\\_set\\_field](#) (struct [ds\\_record](#) \*record, const size\_t index, [ds\\_str](#) field)
- [ds\\_str](#) [ds\\_record\\_get\\_field](#) (struct [ds\\_record](#) \*record, const size\_t index)
- size\_t [ds\\_record\\_size](#) (struct [ds\\_record](#) \*record)
- void [ds\\_record\\_seek\\_start](#) (struct [ds\\_record](#) \*record)
- [ds\\_str](#) [ds\\_record\\_get\\_next\\_data](#) (struct [ds\\_record](#) \*record)
- [ds\\_record](#) [ds\\_record\\_tokenize](#) ([ds\\_str](#) str, const char delim)  
*Tokenizes a string into a record.*
- [ds\\_str](#) [ds\\_record\\_make\\_delim\\_string](#) ([ds\\_record](#) record, const char delim)  
*Makes a delimited string from a record.*
- [ds\\_str](#) [ds\\_record\\_make\\_values\\_string](#) ([ds\\_record](#) record)  
*Makes a delimited SQL values string from a record.*

### 4.37.1 Detailed Description

Implementation of record database structure.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.37.2 Function Documentation

#### 4.37.2.1 struct [ds\\_record](#)\* [ds\\_record\\_create](#) ( const size\_t *size* ) [read]

Creates a new record.

#### Parameters

<i>size</i>	The size of the record.
-------------	-------------------------

#### Returns

A newly created record, or NULL on failure.

#### 4.37.2.2 void ds\_record\_destructor ( void \* *record* )

A record destructor function.

##### Parameters

<i>record</i>	The record to destroy.
---------------	------------------------

#### 4.37.2.3 ds\_str ds\_record\_make\_delim\_string ( ds\_record *record*, const char *delim* )

Makes a delimited string from a record.

##### Parameters

<i>record</i>	The record.
<i>delim</i>	The delimiting character.

##### Returns

The delimited string, or `NULL` on failure.

#### 4.37.2.4 ds\_str ds\_record\_make\_values\_string ( ds\_record *record* )

Makes a delimited SQL values string from a record.

##### Parameters

<i>record</i>	The record.
---------------	-------------

##### Returns

The delimited values string, or `NULL` on failure.

#### 4.37.2.5 ds\_record ds\_record\_tokenize ( ds\_str *str*, const char *delim* )

Tokenizes a string into a record.

##### Parameters

<i>str</i>	The string to tokenize.
<i>delim</i>	The delimiting character.

##### Returns

A new record containing the tokens.

## 4.38 lib/datastruct/ds\_record.h File Reference

Interface to record data structure.

```
#include <stddef.h>
#include <stdbool.h>
#include "ds_str.h"
```





*Returns the size of a record.*

- void [ds\\_record\\_seek\\_start](#) ([ds\\_record](#) record)

*Sets the current field to the first field of a record.*

- [ds\\_str](#) [ds\\_record\\_get\\_next\\_data](#) ([ds\\_record](#) record)

*Returns the next field of the record.*

- [ds\\_record](#) [ds\\_record\\_tokenize](#) ([ds\\_str](#) str, const char delim)

*Tokenizes a string into a record.*

- [ds\\_str](#) [ds\\_record\\_make\\_delim\\_string](#) ([ds\\_record](#) record, const char delim)

*Makes a delimited string from a record.*

- [ds\\_str](#) [ds\\_record\\_make\\_values\\_string](#) ([ds\\_record](#) record)

*Makes a delimited SQL values string from a record.*

### 4.38.1 Detailed Description

Interface to record data structure.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.38.2 Typedef Documentation

#### 4.38.2.1 typedef struct [ds\\_record](#)\* [ds\\_record](#)

Typedef for opaque record datatype

### 4.38.3 Function Documentation

#### 4.38.3.1 void [ds\\_record\\_clear](#) ( [ds\\_record](#) record )

Clears and `free()`s all the elements in a record.

#### Parameters

<i>record</i>	The record.
---------------	-------------

#### 4.38.3.2 [ds\\_record](#) [ds\\_record\\_create](#) ( const [size\\_t](#) size ) [read]

Creates a new record.

#### Parameters

<i>size</i>	The size of the record.
-------------	-------------------------

#### Returns

A newly created record, or `NULL` on failure.

#### 4.38.3.3 void ds\_record\_destroy ( ds\_record record )

Destroys a record and frees any associated resources.

##### Parameters

<i>record</i>	The record to destroy.
---------------	------------------------

#### 4.38.3.4 void ds\_record\_destructor ( void \* record )

A record destructor function.

##### Parameters

<i>record</i>	The record to destroy.
---------------	------------------------

#### 4.38.3.5 ds\_str ds\_record\_get\_field ( ds\_record record, const size\_t index )

Retrieves the field at a specified index.

##### Parameters

<i>record</i>	The record from which to retrieve.
<i>index</i>	The index of the desired field.

##### Returns

A pointer to the field, or `NULL` if the index is out of range.

#### 4.38.3.6 ds\_str ds\_record\_get\_next\_data ( ds\_record record )

Returns the next field of the record.

This function returns the data of the "current field", and advances the current field pointer. Subsequent calls to this function will return successive fields.

##### Parameters

<i>record</i>	The record.
---------------	-------------

##### Returns

A pointer to the next field, or `NULL` if the end of the record has been reached.

#### 4.38.3.7 ds\_str ds\_record\_make\_delim\_string ( ds\_record record, const char delim )

Makes a delimited string from a record.

##### Parameters

<i>record</i>	The record.
<i>delim</i>	The delimiting character.

## Returns

The delimited string, or `NULL` on failure.

4.38.3.8 `ds_str ds_record_make_values_string ( ds_record record )`

Makes a delimited SQL values string from a record.

## Parameters

<i>record</i>	The record.
---------------	-------------

## Returns

The delimited values string, or `NULL` on failure.

4.38.3.9 `void ds_record_seek_start ( ds_record record )`

Sets the current field to the first field of a record.

## Parameters

<i>record</i>	The record.
---------------	-------------

4.38.3.10 `void ds_record_set_field ( ds_record record, const size_t index, ds_str field )`

Sets a field of a record.

If the field is currently occupied, the existing field is `free()`d.

## Parameters

<i>field</i>	The field to which to set.
<i>element</i>	The element to set.

4.38.3.11 `size_t ds_record_size ( ds_record record )`

Returns the size of a record.

## Parameters

<i>record</i>	The record.
---------------	-------------

## Returns

The size of the record.

4.38.3.12 `ds_record ds_record_tokenize ( ds_str str, const char delim )`

Tokenizes a string into a record.

## Parameters

<i>str</i>	The string to tokenize.
<i>delim</i>	The delimiting character.

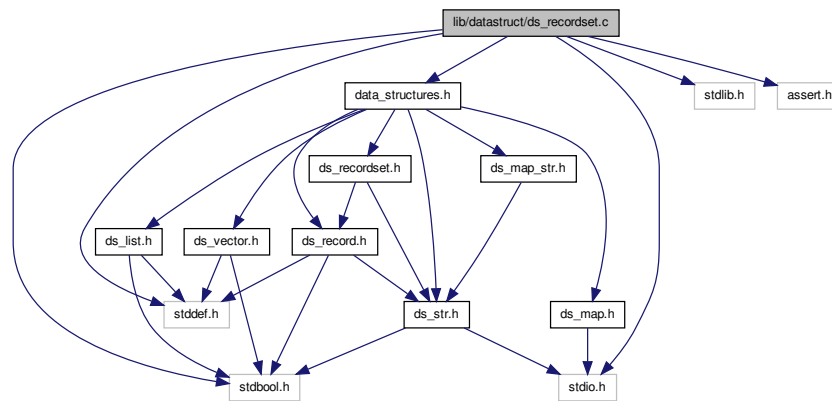
## Returns

A new record containing the tokens.

## 4.39 lib/datastruct/ds\_recordset.c File Reference

Implementation of query result set structure.

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <stdbool.h>
#include <assert.h>
#include "data_structures.h"
Include dependency graph for ds_recordset.c:
```



## Data Structures

- struct [ds\\_recordset](#)

## Functions

- struct [ds\\_recordset](#) \* [ds\\_recordset\\_create](#) (const size\_t num\_fields)  
*Creates a new record set.*
- void [ds\\_recordset\\_destroy](#) (struct [ds\\_recordset](#) \*set)
- [ds\\_record](#) [ds\\_recordset\\_add\\_record](#) (struct [ds\\_recordset](#) \*set, [ds\\_record](#) record)
- size\_t [ds\\_recordset\\_num\\_fields](#) (struct [ds\\_recordset](#) \*set)
- size\_t [ds\\_recordset\\_num\\_records](#) (struct [ds\\_recordset](#) \*set)
- void [ds\\_recordset\\_set\\_headers](#) (struct [ds\\_recordset](#) \*set, [ds\\_record](#) headers)
- [ds\\_str](#) [ds\\_recordset\\_get\\_text\\_report](#) (struct [ds\\_recordset](#) \*set)
- void [ds\\_recordset\\_seek\\_start](#) (struct [ds\\_recordset](#) \*set)
- [ds\\_record](#) [ds\\_recordset\\_next\\_record](#) (struct [ds\\_recordset](#) \*set)
- [ds\\_str](#) [ds\\_recordset\\_get\\_next\\_insert\\_query](#) (struct [ds\\_recordset](#) \*set, const char \*table\_name)

### 4.39.1 Detailed Description

Implementation of query result set structure.

## Author

Paul Griffiths

## Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 4.39.2 Function Documentation

## 4.39.2.1 struct ds\_recordset\* ds\_recordset\_create ( const size\_t num\_fields ) [read]

Creates a new record set.

## Parameters

<i>num_fields</i>	The non-zero number of fields in the record set.
-------------------	--

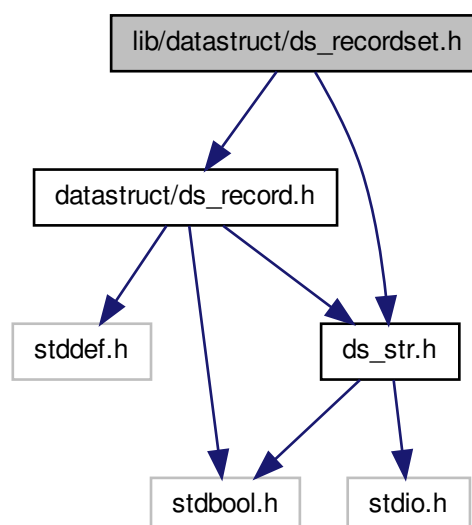
## Returns

A pointer to the new record set.

## 4.40 lib/datastruct/ds\_recordset.h File Reference

Interface to record set structure.

```
#include "datastruct/ds_record.h"
#include "datastruct/ds_str.h"
Include dependency graph for ds_recordset.h:
```





## 4.40.2 Typedef Documentation

### 4.40.2.1 typedef struct ds\_recordset\* ds\_recordset

Typedef for opaque record set data type

## 4.40.3 Function Documentation

### 4.40.3.1 ds\_record ds\_recordset\_add\_record ( ds\_recordset set, ds\_record record )

Adds a record to a record set.

The record *must* have the same number of fields as the number of fields provided to [ds\\_recordset\\_create\(\)](#).

#### Parameters

<i>set</i>	The record set to which to add.
<i>record</i>	The record to add.

#### Returns

A pointer to the new record (i.e. it returns the second parameter) or `NULL` on failure.

### 4.40.3.2 ds\_recordset ds\_recordset\_create ( const size\_t num\_fields ) [read]

Creates a new record set.

#### Parameters

<i>num_fields</i>	The non-zero number of fields in the record set.
-------------------	--

#### Returns

A pointer to the new record set.

### 4.40.3.3 void ds\_recordset\_destroy ( ds\_recordset set )

Destroys a record set and frees associated resources.

#### Parameters

<i>set</i>	The record set to destroy.
------------	----------------------------

### 4.40.3.4 ds\_str ds\_recordset\_get\_next\_insert\_query ( ds\_recordset set, const char \* table\_name )

Gets the next SQL INSERT query.

#### Parameters

<i>set</i>	The set.
<i>table_name</i>	The table name into which to insert.

**Returns**

The query. Caller is responsible for `free()` ing.

**4.40.3.5 `ds_str ds_recordset_get_text_report ( ds_recordset set )`**

Returns a formatted text report for the record set.

The report is returned as a single multi-line string.

**Parameters**

<i>set</i>	The record set.
------------	-----------------

**Returns**

A pointer to the report. The caller is responsible for `free()` ing this pointer.

**4.40.3.6 `ds_record ds_recordset_next_record ( ds_recordset set )`**

Returns the next record in the record set.

This function returns the "current record", and advances the current record pointer. Subsequent calls to this function will return successive records.

**Parameters**

<i>list</i>	The record set.
-------------	-----------------

**Returns**

A pointer to the next record, or `NULL` if the end of the record set has been reached.

**4.40.3.7 `size_t ds_recordset_num_fields ( ds_recordset set )`**

Returns the number of fields in a record set.

**Parameters**

<i>set</i>	The record set.
------------	-----------------

**Returns**

The number of fields in the record set.

**4.40.3.8 `size_t ds_recordset_num_records ( ds_recordset set )`**

Returns the number of records in a record set.

**Parameters**

<i>set</i>	The record set.
------------	-----------------



## Returns

The number of records in the record set.

## 4.40.3.9 void ds\_recordset\_seek\_start ( ds\_recordset set )

Sets the current record to the first record.

## Parameters

<i>set</i>	The record set.
------------	-----------------

## 4.40.3.10 void ds\_recordset\_set\_headers ( ds\_recordset set, ds\_record headers )

Sets the record headers in a record set.

## Parameters

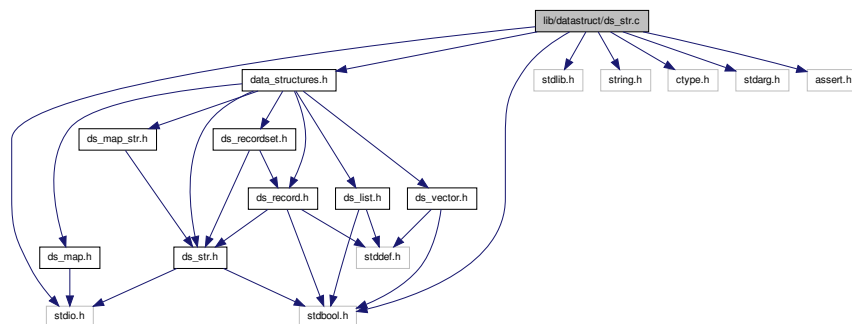
<i>set</i>	The record set.
<i>headers</i>	The headers, in the form of a <a href="#">ds_record</a> of strings. The list <i>must</i> have the same number of elements as the number of fields provided to <a href="#">ds_recordset_create()</a> .

## 4.41 lib/datastruct/ds\_str.c File Reference

Implementation of string data structure.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <ctype.h>
#include <stdarg.h>
#include <assert.h>
#include "data_structures.h"
```

Include dependency graph for ds\_str.c:



## Data Structures

- struct [ds\\_str](#)

## Functions

- [ds\\_str ds\\_str\\_create\\_direct](#) (char \*init\_str, const size\_t init\_str\_size)  
*Creates a string using allocated memory.*
- [ds\\_str ds\\_str\\_create](#) (const char \*init\_str)  
*Creates a new string from a C-style string.*
- [ds\\_str ds\\_str\\_dup](#) (ds\_str src)  
*Creates a new string from another string.*
- [ds\\_str ds\\_str\\_create\\_sprintf](#) (const char \*format,...)  
*Creates a string with `sprintf()`-type format.*
- void [ds\\_str\\_destroy](#) (ds\_str str)  
*Destroys a string and releases allocated resources.*
- void [ds\\_str\\_destructor](#) (void \*str)  
*Destroys a string and releases allocated resources.*
- [ds\\_str ds\\_str\\_assign](#) (ds\_str dst, ds\_str src)  
*Assigns a string to another.*
- [ds\\_str ds\\_str\\_assign\\_cstr](#) (ds\_str dst, const char \*src)  
*Assigns a C-style string to a string.*
- const char \* [ds\\_str\\_cstr](#) (ds\_str str)  
*Returns a C-style string containing the string's contents.*
- size\_t [ds\\_str\\_length](#) (ds\_str str)  
*Returns the length of a string.*
- [ds\\_str ds\\_str\\_size\\_to\\_fit](#) (ds\_str str)
- [ds\\_str ds\\_str\\_concat](#) (ds\_str dst, ds\_str src)  
*Concatenates two strings.*
- [ds\\_str ds\\_str\\_concat\\_cstr](#) (ds\_str dst, const char \*src)
- [ds\\_str ds\\_str\\_trunc](#) (ds\_str str, const size\_t length)  
*Truncates a string.*
- unsigned long [ds\\_str\\_hash](#) (ds\_str str)
- int [ds\\_str\\_compare](#) (ds\_str s1, ds\_str s2)  
*Compares two strings.*
- int [ds\\_str\\_compare\\_cstr](#) (ds\_str s1, const char \*s2)  
*Compares a string with a C-style string.*
- int [ds\\_str\\_strchr](#) (ds\_str str, const char ch, const int start)  
*Returns index of first occurrence of a character.*
- [ds\\_str ds\\_str\\_substr\\_left](#) (ds\_str str, const size\_t numchars)  
*Returns a left substring.*
- [ds\\_str ds\\_str\\_substr\\_right](#) (ds\_str str, const size\_t numchars)  
*Returns a right substring.*
- void [ds\\_str\\_split](#) (ds\_str src, ds\_str \*left, ds\_str \*right, const char sc)  
*Splits a string.*
- void [ds\\_str\\_trim\\_leading](#) (ds\_str str)  
*Trims leading whitespace in-place.*
- void [ds\\_str\\_trim\\_trailing](#) (ds\_str str)  
*Trims trailing whitespace in-place.*
- void [ds\\_str\\_trim](#) (ds\_str str)  
*Trims leading and trailing whitespace in-place.*
- char [ds\\_str\\_char\\_at\\_index](#) (ds\_str str, const size\_t index)  
*Returns the character at a specified index.*
- bool [ds\\_str\\_is\\_empty](#) (ds\_str str)  
*Checks if a string is empty.*

- void `ds_str_clear` (`ds_str` str)  
*Clears (empties) a string.*
- bool `ds_str_intval` (`ds_str` str, const int base, int \*value)  
*Gets the integer value of a string.*
- bool `ds_str_doubleval` (`ds_str` str, double \*value)  
*Gets the double value of a string.*
- `ds_str ds_str_getline` (`ds_str` str, const size\_t size, FILE \*fp)  
*Gets a line from a file and assigns it to a string.*
- `ds_str ds_str_decorate` (`ds_str` str, `ds_str` left\_dec, `ds_str` right\_dec)  
*Brackets a string with decoration strings.*

#### 4.41.1 Detailed Description

Implementation of string data structure.

##### Author

Paul Griffiths

##### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

#### 4.41.2 Function Documentation

##### 4.41.2.1 `ds_str ds_str_assign ( ds_str dst, ds_str src )`

Assigns a string to another.

##### Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source string.

##### Returns

*dst* on success, NULL on failure.

##### 4.41.2.2 `ds_str ds_str_assign_cstr ( ds_str dst, const char * src )`

Assigns a C-style string to a string.

##### Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source C-style string.

##### Returns

*dst* on success, NULL on failure.

#### 4.41.2.3 `char ds_str_char_at_index ( ds_str str, const size_t index )`

Returns the character at a specified index.

##### Parameters

<i>str</i>	The string.
<i>index</i>	The specified index.

##### Returns

The character at the specified index.

#### 4.41.2.4 `void ds_str_clear ( ds_str str )`

Clears (empties) a string.

##### Parameters

<i>str</i>	The string.
------------	-------------

#### 4.41.2.5 `int ds_str_compare ( ds_str s1, ds_str s2 )`

Compares two strings.

##### Parameters

<i>s1</i>	The first string.
<i>s2</i>	The second string.

##### Returns

Less than, equal to, or greater than zero if s1 is found, respectively, to be less than, equal to, or greater than s2.

#### 4.41.2.6 `int ds_str_compare_cstr ( ds_str s1, const char * s2 )`

Compares a string with a C-style string.

##### Parameters

<i>s1</i>	The first string.
<i>s2</i>	The second, C-Style string.

##### Returns

Less than, equal to, or greater than zero if s1 is found, respectively, to be less than, equal to, or greater than s2.

#### 4.41.2.7 `ds_str ds_str_concat ( ds_str dst, ds_str src )`

Concatenates two strings.

## Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source strings.

## Returns

The destination string, or `NULL` on failure.

4.41.2.8 `ds_str ds_str_create ( const char * init_str )`

Creates a new string from a C-style string.

## Parameters

<i>init_str</i>	The C-style string.
-----------------	---------------------

## Returns

The new string, or `NULL` on failure.

4.41.2.9 `ds_str ds_str_create_direct ( char * init_str, const size_t init_str_size )`

Creates a string using allocated memory.

The normal construction functions duplicate the string used to create it. In cases where allocated memory is already available (e.g. in `ds_str_create_sprintf()`) this function allows that memory to be directly assigned to the string, avoiding an unnecessary duplication.

## Parameters

<i>str</i>	The allocated memory. IMPORTANT: If the construction of the string fails, this memory will be <code>free()</code> d.
<i>init_str_size</i>	The size of the allocated memory. IMPORTANT: The string's length is assumed to be one less than this quantity, and a call to <code>strlen()</code> is NOT performed.

## Returns

The new string, or `NULL` on failure.

4.41.2.10 `ds_str ds_str_create_sprintf ( const char * format, ... )`

Creates a string with `sprintf()`-type format.

## Parameters

<i>format</i>	The format string.
<i>...</i>	The subsequent arguments as specified by the format string.

## Returns

The new string, or `NULL` on failure.

#### 4.41.2.11 `const char* ds_str_cstr ( ds_str str )`

Returns a C-style string containing the string's contents.

##### Parameters

<i>str</i>	The string.
------------	-------------

##### Returns

The C-style string containing the string's contents. The caller should not directly modify this string.

#### 4.41.2.12 `ds_str ds_str_decorate ( ds_str str, ds_str left_dec, ds_str right_dec )`

Brackets a string with decoration strings.

##### Parameters

<i>str</i>	The string to decorate.
<i>left_dec</i>	The string to add to the left of <i>str</i> .
<i>right_dec</i>	The string to add to the right of <i>str</i> , or NULL to add <i>left_dec</i> to both sides.

##### Returns

The decorated string.

#### 4.41.2.13 `void ds_str_destroy ( ds_str str )`

Destroys a string and releases allocated resources.

##### Parameters

<i>str</i>	The string to destroy..
------------	-------------------------

#### 4.41.2.14 `void ds_str_destructor ( void * str )`

Destroys a string and releases allocated resources.

This function calls `ds_str_destroy()`, and can be passed to a data structure expecting a destructor function with the signature `void (*)(void *)`.

##### Parameters

<i>str</i>	The string to destroy.
------------	------------------------

#### 4.41.2.15 `bool ds_str_doubleval ( ds_str str, double * value )`

Gets the double value of a string.

##### Parameters

<i>str</i>	The string.
<i>value</i>	A pointer to the double in which to store the value. Zero is stored if the string does not contain a valid double value.

**Returns**

`true` on successful conversion, `false` if the string does not contain a valid double value.

**4.41.2.16 ds\_str ds\_str\_dup ( ds\_str src )**

Creates a new string from another string.

**Parameters**

<i>src</i>	The other string.
------------	-------------------

**Returns**

The new string, or `NULL` on failure.

**4.41.2.17 ds\_str ds\_str\_getline ( ds\_str str, const size\_t size, FILE \* fp )**

Gets a line from a file and assigns it to a string.

Any trailing newline character is stripped.

**Parameters**

<i>str</i>	The string.
<i>size</i>	The maximum number of bytes to read, including the null.
<i>fp</i>	The file pointer from which to read.

**Returns**

*dst*

**4.41.2.18 bool ds\_str\_intval ( ds\_str str, const int base, int \* value )**

Gets the integer value of a string.

**Parameters**

<i>str</i>	The string.
<i>base</i>	The base of the integer. This has the same meaning as the third argument to standard C <code>strtol()</code> .
<i>value</i>	A pointer to the integer in which to store the value. Zero is stored if the string does not contain a valid integer value.

**Returns**

`true` on successful conversion, `false` if the string does not contain a valid integer value.

**4.41.2.19 bool ds\_str\_is\_empty ( ds\_str str )**

Checks if a string is empty.

## Parameters

<i>str</i>	The string.
------------	-------------

## Returns

`true` if the string is empty, `false` otherwise.

4.41.2.20 `size_t ds_str_length ( ds_str str )`

Returns the length of a string.

## Parameters

<i>str</i>	The string.
------------	-------------

## Returns

The length of the string.

4.41.2.21 `void ds_str_split ( ds_str src, ds_str * left, ds_str * right, const char sc )`

Splits a string.

## Parameters

<i>src</i>	The string to split.
<i>left</i>	Pointer to left substring (modified)
<i>right</i>	Pointer to right substring (modified)
<i>sc</i>	Split character.

4.41.2.22 `int ds_str_strchr ( ds_str str, const char ch, const int start )`

Returns index of first occurrence of a character.

## Parameters

<i>str</i>	The string.
<i>ch</i>	The character for which to search.
<i>start</i>	The index of the string at which to start looking. Set this to non-zero to begin searching from a point other than the first character of the string.

## Returns

The index of the first occurrence, or -1 if the character was not found.

4.41.2.23 `ds_str ds_str_substr_left ( ds_str str, const size_t numchars )`

Returns a left substring.

## Parameters

<i>str</i>	The string.
<i>numchars</i>	The number of left characters to return. If this is greater than the length of the string, the whole string is returned.



## Returns

A new string representing the substring.

4.41.2.24 `ds_str ds_str_substr_right ( ds_str str, const size_t numchars )`

Returns a right substring.

## Parameters

<i>str</i>	The string.
<i>numchars</i>	The number of right characters to return. If this is greater than the length of the string, the whole string is returned.

## Returns

A new string representing the substring.

4.41.2.25 `void ds_str_trim ( ds_str str )`

Trims leading and trailing whitespace in-place.

## Parameters

<i>str</i>	The string.
------------	-------------

4.41.2.26 `void ds_str_trim_leading ( ds_str str )`

Trims leading whitespace in-place.

## Parameters

<i>str</i>	The string.
------------	-------------

4.41.2.27 `void ds_str_trim_trailing ( ds_str str )`

Trims trailing whitespace in-place.

## Parameters

<i>str</i>	The string.
------------	-------------

4.41.2.28 `ds_str ds_str_trunc ( ds_str str, const size_t length )`

Truncates a string.

## Parameters

<i>str</i>	The string.
<i>length</i>	The new length to which to truncate.



- Creates a string using allocated memory.*
- void [ds\\_str\\_destroy](#) ([ds\\_str](#) str)
- Destroys a string and releases allocated resources.*
- void [ds\\_str\\_destructor](#) (void \*str)
- Destroys a string and releases allocated resources.*
- [ds\\_str ds\\_str\\_assign](#) ([ds\\_str](#) dst, [ds\\_str](#) src)
- Assigns a string to another.*
- [ds\\_str ds\\_str\\_assign\\_cstr](#) ([ds\\_str](#) dst, const char \*src)
- Assigns a C-style string to a string.*
- const char \* [ds\\_str\\_cstr](#) ([ds\\_str](#) str)
- Returns a C-style string containing the string's contents.*
- size\_t [ds\\_str\\_length](#) ([ds\\_str](#) str)
- Returns the length of a string.*
- [ds\\_str ds\\_str\\_concat](#) ([ds\\_str](#) dst, [ds\\_str](#) src)
- Concatenates two strings.*
- [ds\\_str ds\\_str\\_concat\\_cstr](#) (struct [ds\\_str](#) \*dst, const char \*src)
- Concatenates a C-style string to a string.*
- [ds\\_str ds\\_str\\_trunc](#) ([ds\\_str](#) str, const size\_t length)
- Truncates a string.*
- unsigned long [ds\\_str\\_hash](#) (struct [ds\\_str](#) \*str)
- Calculates a hash of a string.*
- int [ds\\_str\\_compare](#) ([ds\\_str](#) s1, [ds\\_str](#) s2)
- Compares two strings.*
- int [ds\\_str\\_compare\\_cstr](#) ([ds\\_str](#) s1, const char \*s2)
- Compares a string with a C-style string.*
- int [ds\\_str\\_strchr](#) ([ds\\_str](#) str, const char ch, const int start)
- Returns index of first occurrence of a character.*
- [ds\\_str ds\\_str\\_substr\\_left](#) ([ds\\_str](#) str, const size\_t numchars)
- Returns a left substring.*
- [ds\\_str ds\\_str\\_substr\\_right](#) ([ds\\_str](#) str, const size\_t numchars)
- Returns a right substring.*
- void [ds\\_str\\_split](#) ([ds\\_str](#) src, [ds\\_str](#) \*left, [ds\\_str](#) \*right, const char sc)
- Splits a string.*
- void [ds\\_str\\_trim\\_leading](#) ([ds\\_str](#) str)
- Trims leading whitespace in-place.*
- void [ds\\_str\\_trim\\_trailing](#) ([ds\\_str](#) str)
- Trims trailing whitespace in-place.*
- void [ds\\_str\\_trim](#) ([ds\\_str](#) str)
- Trims leading and trailing whitespace in-place.*
- char [ds\\_str\\_char\\_at\\_index](#) ([ds\\_str](#) str, const size\_t index)
- Returns the character at a specified index.*
- bool [ds\\_str\\_is\\_empty](#) ([ds\\_str](#) str)
- Checks if a string is empty.*
- void [ds\\_str\\_clear](#) ([ds\\_str](#) str)
- Clears (empties) a string.*
- bool [ds\\_str\\_intval](#) ([ds\\_str](#) str, const int base, int \*value)
- Gets the integer value of a string.*
- bool [ds\\_str\\_doubleval](#) ([ds\\_str](#) str, double \*value)
- Gets the double value of a string.*
- [ds\\_str ds\\_str\\_getline](#) ([ds\\_str](#) str, const size\_t size, FILE \*fp)
- Gets a line from a file and assigns it to a string.*
- [ds\\_str ds\\_str\\_decorate](#) ([ds\\_str](#) str, [ds\\_str](#) left\_dec, [ds\\_str](#) right\_dec)
- Brackets a string with decoration strings.*

### 4.42.1 Detailed Description

Interface to string data structure.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.42.2 Typedef Documentation

#### 4.42.2.1 typedef struct **ds\_str**\* **ds\_str**

Opaque data type for string

### 4.42.3 Function Documentation

#### 4.42.3.1 **ds\_str** **ds\_str\_assign** ( **ds\_str** *dst*, **ds\_str** *src* )

Assigns a string to another.

#### Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source string.

#### Returns

*dst* on success, NULL on failure.

#### 4.42.3.2 **ds\_str** **ds\_str\_assign\_cstr** ( **ds\_str** *dst*, const char \* *src* )

Assigns a C-style string to a string.

#### Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source C-style string.

#### Returns

*dst* on success, NULL on failure.

#### 4.42.3.3 char **ds\_str\_char\_at\_index** ( **ds\_str** *str*, const size\_t *index* )

Returns the character at a specified index.

#### Parameters

<i>str</i>	The string.
<i>index</i>	The specified index.

**Returns**

The character at the specified index.

**4.42.3.4 void ds\_str\_clear ( ds\_str str )**

Clears (empties) a string.

**Parameters**

<i>str</i>	The string.
------------	-------------

**4.42.3.5 int ds\_str\_compare ( ds\_str s1, ds\_str s2 )**

Compares two strings.

**Parameters**

<i>s1</i>	The first string.
<i>s2</i>	The second string.

**Returns**

Less than, equal to, or greater than zero if *s1* is found, respectively, to be less than, equal to, or greater than *s2*.

**4.42.3.6 int ds\_str\_compare\_cstr ( ds\_str s1, const char \* s2 )**

Compares a string with a C-style string.

**Parameters**

<i>s1</i>	The first string.
<i>s2</i>	The second, C-Style string.

**Returns**

Less than, equal to, or greater than zero if *s1* is found, respectively, to be less than, equal to, or greater than *s2*.

**4.42.3.7 ds\_str ds\_str\_concat ( ds\_str dst, ds\_str src )**

Concatenates two strings.

**Parameters**

<i>dst</i>	The destination string.
<i>src</i>	The source strings.

**Returns**

The destination string, or `NULL` on failure.

#### 4.42.3.8 `ds_str ds_str_concat_cstr ( struct ds_str * dst, const char * src )`

Concatenates a C-style string to a string.

##### Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source strings.

##### Returns

The destination string, or `NULL` on failure.

#### 4.42.3.9 `ds_str ds_str_create ( const char * init_str )`

Creates a new string from a C-style string.

##### Parameters

<i>init_str</i>	The C-style string.
-----------------	---------------------

##### Returns

The new string, or `NULL` on failure.

#### 4.42.3.10 `ds_str ds_str_create_direct ( char * init_str, const size_t init_str_size )`

Creates a string using allocated memory.

The normal construction functions duplicate the string used to create it. In cases where allocated memory is already available (e.g. in `ds_str_create_sprintf()`) this function allows that memory to be directly assigned to the string, avoiding an unnecessary duplication.

##### Parameters

<i>str</i>	The allocated memory. IMPORTANT: If the construction of the string fails, this memory will be <code>free()</code> d.
<i>init_str_size</i>	The size of the allocated memory. IMPORTANT: The string's length is assumed to be one less than this quantity, and a call to <code>strlen()</code> is NOT performed.

##### Returns

The new string, or `NULL` on failure.

#### 4.42.3.11 `ds_str ds_str_create_sprintf ( const char * format, ... )`

Creates a string with `sprintf()`-type format.

##### Parameters

<i>format</i>	The format string.
<i>...</i>	The subsequent arguments as specified by the format string.

**Returns**

The new string, or `NULL` on failure.

**4.42.3.12 `const char* ds_str_cstr ( ds_str str )`**

Returns a C-style string containing the string's contents.

**Parameters**

<i>str</i>	The string.
------------	-------------

**Returns**

The C-style string containing the string's contents. The caller should not directly modify this string.

**4.42.3.13 `ds_str ds_str_decorate ( ds_str str, ds_str left_dec, ds_str right_dec )`**

Brackets a string with decoration strings.

**Parameters**

<i>str</i>	The string to decorate.
<i>left_dec</i>	The string to add to the left of <i>str</i> .
<i>right_dec</i>	The string to add to the right of <i>str</i> , or <code>NULL</code> to add <i>left_dec</i> to both sides.

**Returns**

The decorated string.

**4.42.3.14 `void ds_str_destroy ( ds_str str )`**

Destroys a string and releases allocated resources.

**Parameters**

<i>str</i>	The string to destroy..
------------	-------------------------

**4.42.3.15 `void ds_str_destructor ( void * str )`**

Destroys a string and releases allocated resources.

This function calls `ds_str_destroy()`, and can be passed to a data structure expecting a destructor function with the signature `void (*)(void *)`.

**Parameters**

<i>str</i>	The string to destroy.
------------	------------------------

**4.42.3.16 `bool ds_str_doubleval ( ds_str str, double * value )`**

Gets the double value of a string.

## Parameters

<i>str</i>	The string.
<i>value</i>	A pointer to the double in which to store the value. Zero is stored if the string does not contain a valid double value.

## Returns

`true` on successful conversion, `false` if the string does not contain a valid double value.

4.42.3.17 `ds_str ds_str_dup ( ds_str src )`

Creates a new string from another string.

## Parameters

<i>src</i>	The other string.
------------	-------------------

## Returns

The new string, or `NULL` on failure.

4.42.3.18 `ds_str ds_str_getline ( ds_str str, const size_t size, FILE * fp )`

Gets a line from a file and assigns it to a string.

Any trailing newline character is stripped.

## Parameters

<i>str</i>	The string.
<i>size</i>	The maximum number of bytes to read, including the null.
<i>fp</i>	The file pointer from which to read.

## Returns

`dst`

4.42.3.19 `unsigned long ds_str_hash ( struct ds_str * str )`

Calculates a hash of a string.

Uses Dan Bernstein's djb2 algorithm.

## Parameters

<i>str</i>	The string.
------------	-------------

## Returns

The hash value

4.42.3.20 `bool ds_str_intval ( ds_str str, const int base, int * value )`

Gets the integer value of a string.



## Parameters

<i>str</i>	The string.
<i>base</i>	The base of the integer. This has the same meaning as the third argument to standard C <code>strtol()</code> .
<i>value</i>	A pointer to the integer in which to store the value. Zero is stored if the string does not contain a valid integer value.

## Returns

`true` on successful conversion, `false` if the string does not contain a valid integer value.

4.42.3.21 `bool ds_str_is_empty ( ds_str str )`

Checks if a string is empty.

## Parameters

<i>str</i>	The string.
------------	-------------

## Returns

`true` if the string is empty, `false` otherwise.

4.42.3.22 `size_t ds_str_length ( ds_str str )`

Returns the length of a string.

## Parameters

<i>str</i>	The string.
------------	-------------

## Returns

The length of the string.

4.42.3.23 `void ds_str_split ( ds_str src, ds_str * left, ds_str * right, const char sc )`

Splits a string.

## Parameters

<i>src</i>	The string to split.
<i>left</i>	Pointer to left substring (modified)
<i>right</i>	Pointer to right substring (modified)
<i>sc</i>	Split character.

4.42.3.24 `int ds_str_strchr ( ds_str str, const char ch, const int start )`

Returns index of first occurrence of a character.

## Parameters

<i>str</i>	The string.
<i>ch</i>	The character for which to search.
<i>start</i>	The index of the string at which to start looking. Set this to non-zero to begin searching from a point other than the first character of the string.

## Returns

The index of the first occurrence, or -1 if the character was not found.

4.42.3.25 `ds_str ds_str_substr_left ( ds_str str, const size_t numchars )`

Returns a left substring.

## Parameters

<i>str</i>	The string.
<i>numchars</i>	The number of left characters to return. If this is greater than the length of the string, the whole string is returned.

## Returns

A new string representing the substring.

4.42.3.26 `ds_str ds_str_substr_right ( ds_str str, const size_t numchars )`

Returns a right substring.

## Parameters

<i>str</i>	The string.
<i>numchars</i>	The number of right characters to return. If this is greater than the length of the string, the whole string is returned.

## Returns

A new string representing the substring.

4.42.3.27 `void ds_str_trim ( ds_str str )`

Trims leading and trailing whitespace in-place.

## Parameters

<i>str</i>	The string.
------------	-------------

4.42.3.28 `void ds_str_trim_leading ( ds_str str )`

Trims leading whitespace in-place.

## Parameters

<i>str</i>	The string.
------------	-------------

## 4.42.3.29 void ds\_str\_trim\_trailing ( ds\_str str )

Trims trailing whitespace in-place.

## Parameters

<i>str</i>	The string.
------------	-------------

## 4.42.3.30 ds\_str ds\_str\_trunc ( ds\_str str, const size\_t length )

Truncates a string.

## Parameters

<i>str</i>	The string.
<i>length</i>	The new length to which to truncate.

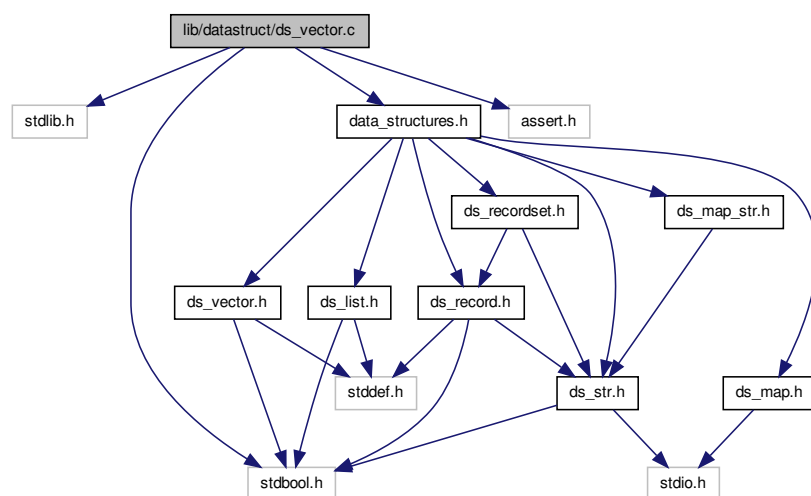
## Returns

The original string, or `NULL` on failure.

## 4.43 lib/datastruct/ds\_vector.c File Reference

Implementation of generic doubly-linked vector data structure.

```
#include <stdlib.h>
#include <stdbool.h>
#include <assert.h>
#include "data_structures.h"
Include dependency graph for ds_vector.c:
```



## Data Structures

- struct [ds\\_vector](#)

## Functions

- struct [ds\\_vector](#) \* [ds\\_vector\\_create](#) (const size\_t size, const bool free\_on\_delete, void(\*destructor)(void \*))  
*Creates a new vector.*
- void [ds\\_vector\\_destroy](#) (struct [ds\\_vector](#) \*vector)
- void [ds\\_vector\\_destructor](#) (void \*vector)  
*A vector destructor function.*
- void [ds\\_vector\\_clear](#) (struct [ds\\_vector](#) \*vector)
- void [ds\\_vector\\_set](#) (struct [ds\\_vector](#) \*vector, const size\_t index, void \*element)
- void \* [ds\\_vector\\_element](#) (struct [ds\\_vector](#) \*vector, const size\_t index)
- size\_t [ds\\_vector\\_size](#) (struct [ds\\_vector](#) \*vector)
- void [ds\\_vector\\_seek\\_start](#) (struct [ds\\_vector](#) \*vector)
- void \* [ds\\_vector\\_get\\_next\\_data](#) (struct [ds\\_vector](#) \*vector)

### 4.43.1 Detailed Description

Implementation of generic doubly-linked vector data structure.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.43.2 Function Documentation

4.43.2.1 struct [ds\\_vector](#)\* [ds\\_vector\\_create](#) ( const size\_t *size*, const bool *free\_on\_delete*, void(\*)*(void \*)* *destructor* )  
[read]

Creates a new vector.

#### Parameters

<i>size</i>	The size of the vector.
<i>free_on_delete</i>	Set to <code>true</code> if the vector elements should be destroyed when removed from the vector, and when the vector itself is destroyed. If set to <code>false</code> , the caller is responsible for destroying the elements prior to destroying the vector.
<i>destructor</i>	Pointer to a destructor function to use for destroying the vector elements, when <code>free_on_delete</code> is true. If this is set to <code>NULL</code> , <code>free()</code> from the standard C library will be used to destroy the elements.

#### Returns

A newly created vector, or `NULL` on failure.

A vector destructor function.

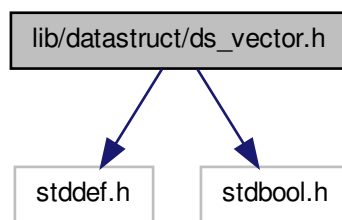
## Parameters

<i>vector</i>	The vector to destroy.
---------------	------------------------

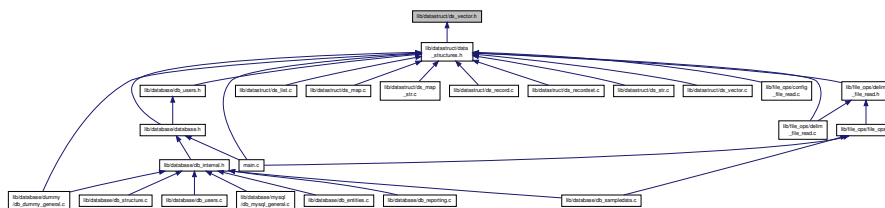
Interface to generic doubly-linked vector data structure.

```
#include <stdbool.h>
```

Include dependency graph for ds\_vector.h:



This graph shows which files directly or indirectly include this file:



- typedef struct ds\_vector \* ds\_vector

- `ds_vector ds_vector_create` (const size\_t size, const bool free\_on\_delete, void(\*destructor)(void \*))  
*Creates a new vector.*
- void `ds_vector_destroy` (ds\_vector vector)

- Destroys a vector and frees any associated resources.*
- void `ds_vector_destructor` (void \*vector)  
*A vector destructor function.*
- void `ds_vector_clear` (ds\_vector vector)  
*Clears all the elements in a vector.*
- void `ds_vector_set` (ds\_vector vector, const size\_t index, void \*element)  
*Sets an element of a vector.*
- void \* `ds_vector_element` (ds\_vector vector, const size\_t index)  
*Retrieves the data at a specified index.*
- size\_t `ds_vector_size` (ds\_vector vector)  
*Returns the size of a vector.*
- void `ds_vector_seek_start` (ds\_vector vector)  
*Sets the current element to the first element of a vector.*
- void \* `ds_vector_get_next_data` (ds\_vector vector)  
*Returns the next element of the vector.*

#### 4.44.1 Detailed Description

Interface to generic doubly-linked vector data structure.

##### Author

Paul Griffiths

##### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

#### 4.44.2 Typedef Documentation

##### 4.44.2.1 typedef struct ds\_vector\* ds\_vector

Typedef for opaque vector datatype

#### 4.44.3 Function Documentation

##### 4.44.3.1 void ds\_vector\_clear ( ds\_vector vector )

Clears all the elements in a vector.

If the vector was created with `free_on_delete`, the elements are `free()`d prior to being cleared (i.e. set to NULL).

##### Parameters

<code>vector</code>	The vector.
---------------------	-------------

##### 4.44.3.2 ds\_vector ds\_vector\_create ( const size\_t size, const bool free\_on\_delete, void(\*) (void \*) destructor ) [read]

Creates a new vector.

## Parameters

<i>size</i>	The size of the vector.
<i>free_on_delete</i>	Set to <code>true</code> if the vector elements should be destroyed when removed from the vector, and when the vector itself is destroyed. If set to <code>false</code> , the caller is responsible for destroying the elements prior to destroying the vector.
<i>destructor</i>	Pointer to a destructor function to use for destroying the vector elements, when <code>free_on_delete</code> is true. If this is set to <code>NULL</code> , <code>free()</code> from the standard C library will be used to destroy the elements.

## Returns

A newly created vector, or `NULL` on failure.

## 4.44.3.3 void ds\_vector\_destroy ( ds\_vector vector )

Destroys a vector and frees any associated resources.

## Parameters

<i>vector</i>	The vector to destroy.
---------------	------------------------

## 4.44.3.4 void ds\_vector\_destructor ( void \* vector )

A vector destructor function.

This function may be passed to `ds_vector_create()` when creating a vector of vectors. It calls `ds_vector_destroy()`, but the parameter of `ds_vector_destroy()` is not compatible with the function signature expected by `ds_vector_create()`, so this function provides an appropriate interface.

## Parameters

<i>vector</i>	The vector to destroy.
---------------	------------------------

## 4.44.3.5 void\* ds\_vector\_element ( ds\_vector vector, const size\_t index )

Retrieves the data at a specified index.

## Parameters

<i>vector</i>	The vector from which to retrieve.
<i>index</i>	The index of the desired element.

## Returns

A pointer to the data, or `NULL` if the index is out of range.

## 4.44.3.6 void\* ds\_vector\_get\_next\_data ( ds\_vector vector )

Returns the next element of the vector.

This function returns the data of the "current element", and advances the current element pointer. Subsequent calls to this function will return successive elements.

## Parameters

<i>vector</i>	The vector.
---------------	-------------

## Returns

A pointer to the next element, or `NULL` if the end of the vector has been reached.

4.44.3.7 `void ds_vector_seek_start ( ds_vector vector )`

Sets the current element to the first element of a vector.

## Parameters

<i>vector</i>	The vector.
---------------	-------------

4.44.3.8 `void ds_vector_set ( ds_vector vector, const size_t index, void * element )`

Sets an element of a vector.

If the element is currently occupied, the existing element is `free()`d.

## Parameters

<i>vector</i>	The vector to which to set.
<i>element</i>	The element to set.

4.44.3.9 `size_t ds_vector_size ( ds_vector vector )`

Returns the size of a vector.

## Parameters

<i>vector</i>	The vector.
---------------	-------------

## Returns

The size of the vector.

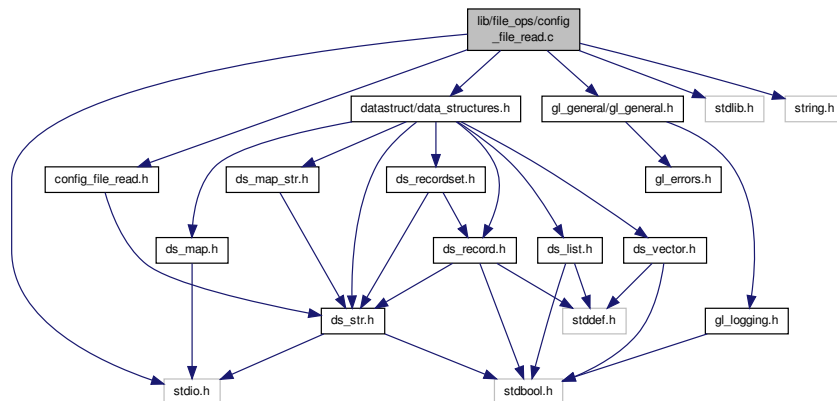
4.45 `lib/file_ops/config_file_read.c` File Reference

Implementation of configuration file reading functionality.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "gl_general/gl_general.h"
#include "datastruct/data_structures.h"
#include "config_file_read.h"
```



Include dependency graph for config\_file\_read.c:



## Macros

- #define [MAX\\_BUFFER\\_SIZE](#) 1024
- #define [CONFIG\\_MAP\\_SIZE](#) 100

## Functions

- int [config\\_file\\_read](#) (const char \*filename)  
*Reads a configuration file and stores the key-value pairs.*
- [ds\\_str config\\_file\\_value](#) (ds\_str key)  
*Returns the value associated with a key.*
- void [config\\_file\\_free](#) (void)  
*Frees the resources used by this module.*

### 4.45.1 Detailed Description

Implementation of configuration file reading functionality. This module reads configuration files in the format "key = value" and makes those values available. Leading and trailing whitespace is removed for both the key and the value. Blank lines and lines starting with a '#' are ignored in the configuration file.

#### Author

Paul Griffiths

#### Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.45.2 Macro Definition Documentation

#### 4.45.2.1 #define CONFIG\_MAP\_SIZE 100

Size to use for the hash map to contain the key-value pairs

#### 4.45.2.2 `#define MAX_BUFFER_SIZE 1024`

Maximum size of buffers

### 4.45.3 Function Documentation

#### 4.45.3.1 `void config_file_free ( void )`

Frees the resources used by this module.

The user should make copies of any required keys or values prior to calling this function. This function need not be called if `config_file_read()` returned an error.

#### 4.45.3.2 `int config_file_read ( const char * filename )`

Reads a configuration file and stores the key-value pairs.

##### Parameters

<i>filename</i>	The name of the configuration file.
-----------------	-------------------------------------

##### Returns

CONFIG\_FILE\_OK on success, CONFIG\_FILE\_NO\_FILE if the specified file could not be opened for reading, CONFIG\_FILE\_MALFORMED\_FILE if the configuration file was improperly formed.

#### 4.45.3.3 `ds_str config_file_value ( ds_str key )`

Returns the value associated with a key.

##### Parameters

<i>key</i>	The specified key.
------------	--------------------

##### Returns

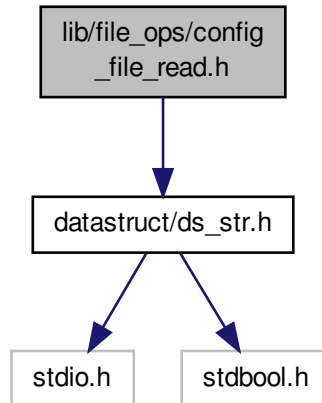
A pointer to the associated value, or `NULL` if the key was not present in the configuration file. The caller should not modify the string to which the pointer points.

## 4.46 `lib/file_ops/config_file_read.h` File Reference

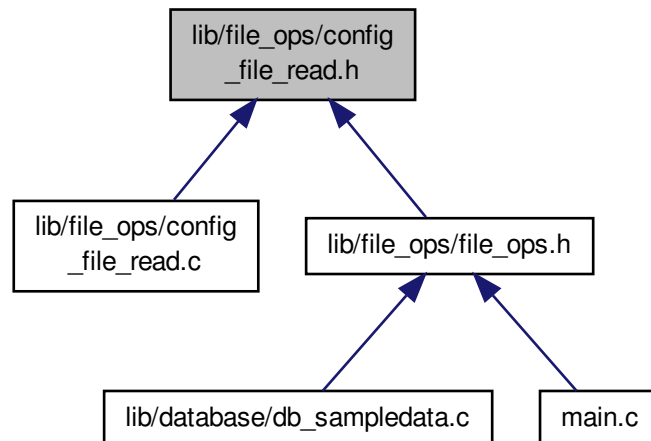
Interface to configuration file reading functionality.

```
#include "datastruct/ds_str.h"
```

Include dependency graph for config\_file\_read.h:



This graph shows which files directly or indirectly include this file:



## Macros

- `#define CONFIG_FILE_OK 0`
- `#define CONFIG_FILE_NO_FILE 1`
- `#define CONFIG_FILE_MALFORMED_FILE 2`

## Functions

- int `config_file_read` (const char \*filename)  
*Reads a configuration file and stores the key-value pairs.*
- void `config_file_free` (void)  
*Frees the resources used by this module.*
- ds\_str `config_file_value` (ds\_str key)  
*Returns the value associated with a key.*

### 4.46.1 Detailed Description

Interface to configuration file reading functionality. This module reads configuration files in the format "key = value" and makes those values available. Leading and trailing whitespace is removed for both the key and the value. Blank lines and lines starting with a '#' are ignored in the configuration file.

#### Author

Paul Griffiths

#### Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.46.2 Macro Definition Documentation

#### 4.46.2.1 #define CONFIG\_FILE\_MALFORMED\_FILE 2

Return status when configuration file is improperly formed

#### 4.46.2.2 #define CONFIG\_FILE\_NO\_FILE 1

Return status when unable to open file for reading

#### 4.46.2.3 #define CONFIG\_FILE\_OK 0

Return status for success

### 4.46.3 Function Documentation

#### 4.46.3.1 void config\_file\_free ( void )

Frees the resources used by this module.

The user should make copies of any required keys or values prior to calling this function. This function need not be called if `config_file_read()` returned an error.

#### 4.46.3.2 int config\_file\_read ( const char \* filename )

Reads a configuration file and stores the key-value pairs.

#### Parameters

<i>filename</i>	The name of the configuration file.
-----------------	-------------------------------------

**Returns**

CONFIG\_FILE\_OK on success, CONFIG\_FILE\_NO\_FILE if the specified file could not be opened for reading, CONFIG\_FILE\_MALFORMED\_FILE if the configuration file was improperly formed.

**4.46.3.3 ds\_str config\_file\_value ( ds\_str key )**

Returns the value associated with a key.

**Parameters**

<i>key</i>	The specified key.
------------	--------------------

**Returns**

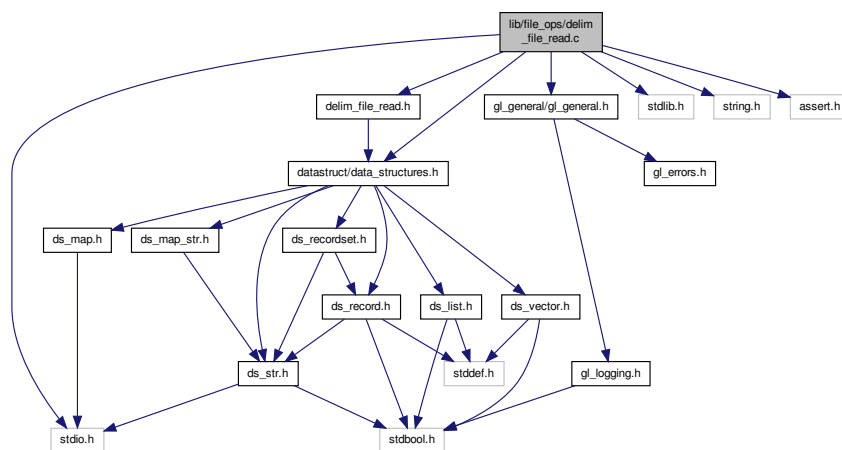
A pointer to the associated value, or NULL if the key was not present in the configuration file. The caller should not modify the string to which the pointer points.

**4.47 lib/file\_ops/delim\_file\_read.c File Reference**

Implementation of delimited file reading functionality.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include "gl_general/gl_general.h"
#include "datastruct/data_structures.h"
#include "delim_file_read.h"
```

Include dependency graph for delim\_file\_read.c:

**Macros**

- #define MAX\_LINE\_SIZE 1024

## Functions

- [ds\\_recordset delim\\_file\\_read](#) (const char \*filename, const char delim)  
Constructs a [ds\\_recordset](#) from a delimited file.

### 4.47.1 Detailed Description

Implementation of delimited file reading functionality.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.47.2 Macro Definition Documentation

#### 4.47.2.1 #define MAX\_LINE\_SIZE 1024

Maximum size of buffers

### 4.47.3 Function Documentation

#### 4.47.3.1 ds\_recordset delim\_file\_read ( const char \* filename, const char delim )

Constructs a [ds\\_recordset](#) from a delimited file.

#### Parameters

<i>filename</i>	The name of the delimited file.
<i>delim</i>	The delimiting character.

#### Returns

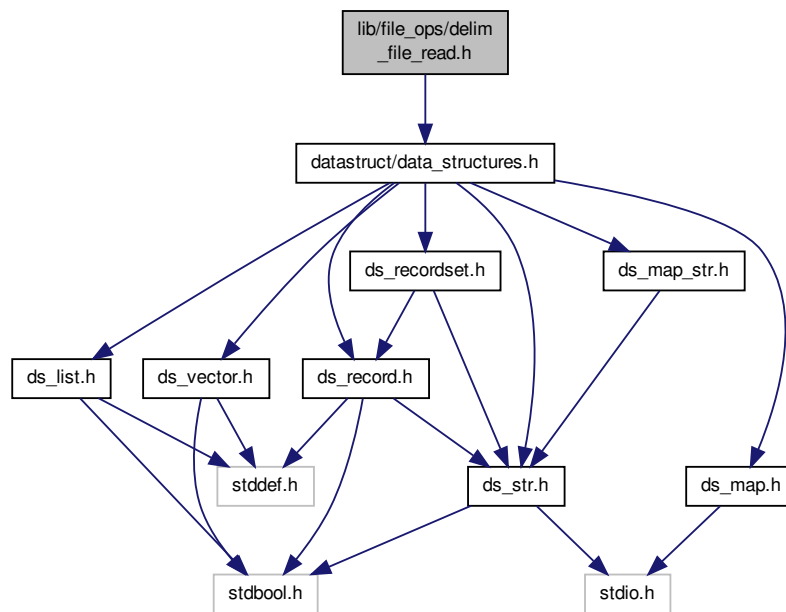
The [ds\\_recordset](#), or NULL on failure.

## 4.48 lib/file\_ops/delim\_file\_read.h File Reference

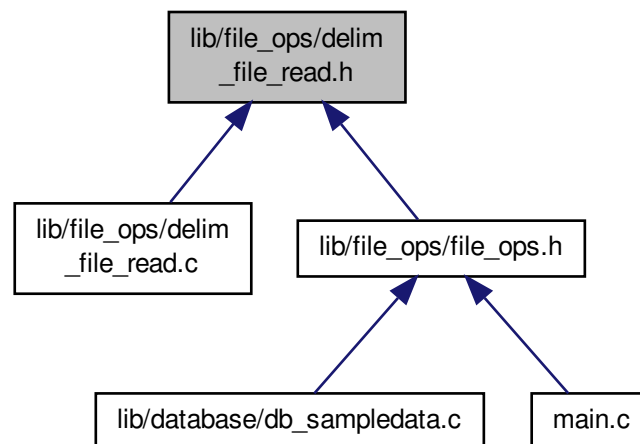
Interface to delimited file reading functionality.

```
#include "datastruct/data_structures.h"
```

Include dependency graph for delim\_file\_read.h:



This graph shows which files directly or indirectly include this file:



## Functions

- [ds\\_recordset delim\\_file\\_read](#) (const char \*filename, const char delim)  
Constructs a [ds\\_recordset](#) from a delimited file.

### 4.48.1 Detailed Description

Interface to delimited file reading functionality.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.48.2 Function Documentation

#### 4.48.2.1 `ds_recordset delim_file_read ( const char * filename, const char delim )`

Constructs a `ds_recordset` from a delimited file.

#### Parameters

<i>filename</i>	The name of the delimited file.
<i>delim</i>	The delimiting character.

#### Returns

The `ds_recordset`, or `NULL` on failure.

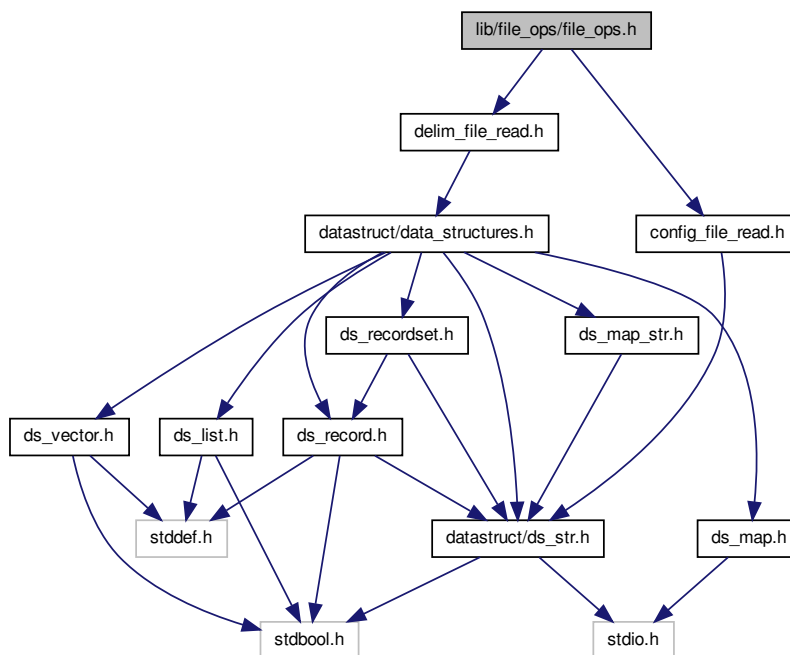
## 4.49 lib/file\_ops/file\_ops.h File Reference

User interface to file operations functionality.

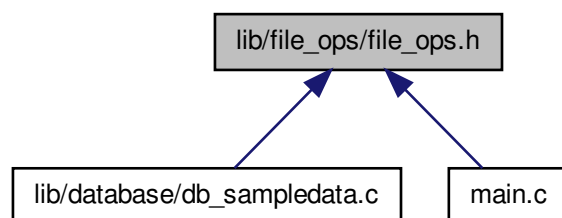
```
#include "config_file_read.h"
#include "delim_file_read.h"
```



Include dependency graph for file\_ops.h:



This graph shows which files directly or indirectly include this file:



#### 4.49.1 Detailed Description

User interface to file operations functionality.

Author

Paul Griffiths

Copyright

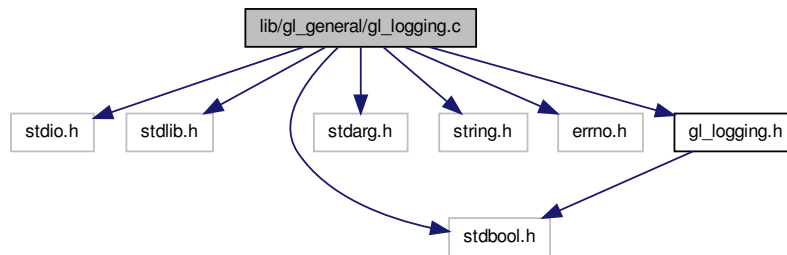
Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 4.50 lib/gl\_general/gl\_logging.c File Reference

Implementation of logging functionality.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <stdarg.h>
#include <string.h>
#include <errno.h>
#include "gl_logging.h"
```

Include dependency graph for gl\_logging.c:



### Functions

- void [gl\\_set\\_logging](#) (const bool status)  
*Turns logging on or off.*
- void [gl\\_log\\_msg](#) (const char \*format,...)  
*Logs a message to the log file.*

#### 4.50.1 Detailed Description

Implementation of logging functionality. Implementation of logging functionality. Enables debugging and other system messages to be recorded to a log file.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

#### 4.50.2 Function Documentation

##### 4.50.2.1 void gl\_log\_msg ( const char \* format, ... )

Logs a message to the log file.

Logs a message to the log file.

## Parameters

<i>format</i>	Format string, in same format as <code>printf()</code> .
...	Variable arguments as specified by format string.

## 4.50.2.2 void gl\_set\_logging ( const bool status )

Turns logging on or off.

Turns logging on or off.

## Parameters

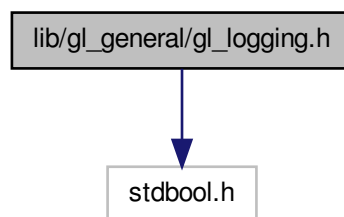
<i>status</i>	true to turn logging on, false to turn logging off.
---------------	---

## 4.51 lib/gl\_general/gl\_logging.h File Reference

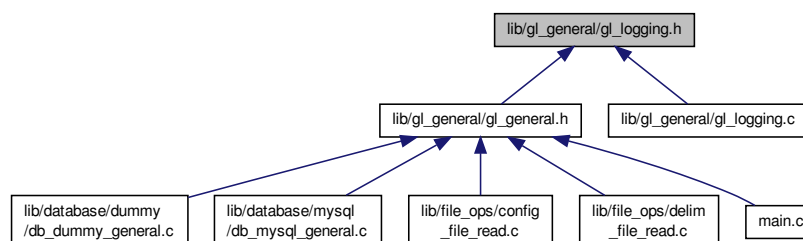
Interface to logging functionality.

```
#include <stdbool.h>
```

Include dependency graph for gl\_logging.h:



This graph shows which files directly or indirectly include this file:



## Functions

- void [gl\\_set\\_logging](#) (const bool status)

*Turns logging on or off.*

- void `gl_log_msg` (const char \*format,...)

*Logs a message to the log file.*

#### 4.51.1 Detailed Description

Interface to logging functionality. Interface to logging functionality. Enables debugging and other system messages to be recorded to a log file.

##### Author

Paul Griffiths

##### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

#### 4.51.2 Function Documentation

##### 4.51.2.1 void `gl_log_msg` ( const char \* *format*, ... )

Logs a message to the log file.

Logs a message to the log file.

##### Parameters

<i>format</i>	Format string, in same format as <code>printf()</code> .
...	Variable arguments as specified by format string.

##### 4.51.2.2 void `gl_set_logging` ( const bool *status* )

Turns logging on or off.

Turns logging on or off.

##### Parameters

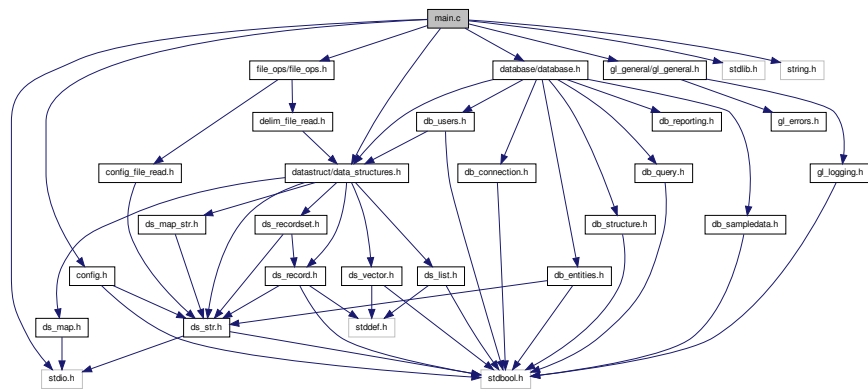
<i>status</i>	true to turn logging on, false to turn logging off.
---------------	---

## 4.52 main.c File Reference

Main function for `general_ledger`.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "gl_general/gl_general.h"
#include "database/database.h"
#include "config.h"
#include "datastruct/data_structures.h"
#include "file_ops/file_ops.h"
```

Include dependency graph for main.c:



## Functions

- **ds\_str login** (void)
- void **print\_usage\_message** (char \*progname)
- void **print\_version\_message** (char \*progname)
- void **print\_help\_message** (char \*progname)
- void **test\_functionality** (void)
- int **main** (int argc, char \*\*argv)

*Main function.*

### 4.52.1 Detailed Description

Main function for general\_ledger. Main function for general\_ledger.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 4.52.2 Function Documentation

#### 4.52.2.1 int main ( int argc, char \*\* argv )

Main function.

Main function.

#### Returns

Exit status.

# Index

- CONFIG\_FILE\_OK
  - config\_file\_read.h, [102](#)
- CONFIG\_MAP\_SIZE
  - config\_file\_read.c, [99](#)
- capacity
  - ds\_str, [13](#)
- config\_file\_free
  - config\_file\_read.c, [100](#)
  - config\_file\_read.h, [102](#)
- config\_file\_read
  - config\_file\_read.c, [100](#)
  - config\_file\_read.h, [102](#)
- config\_file\_read.c
  - CONFIG\_MAP\_SIZE, [99](#)
  - config\_file\_free, [100](#)
  - config\_file\_read, [100](#)
  - config\_file\_value, [100](#)
  - MAX\_BUFFER\_SIZE, [99](#)
- config\_file\_read.h
  - CONFIG\_FILE\_OK, [102](#)
  - config\_file\_free, [102](#)
  - config\_file\_read, [102](#)
  - config\_file\_value, [103](#)
- config\_file\_value
  - config\_file\_read.c, [100](#)
  - config\_file\_read.h, [103](#)
- conn\_mss
  - db\_mysql\_general.c, [46](#)
- current
  - ds\_list, [7](#)
  - ds\_vector, [14](#)
- data
  - ds\_list\_element, [9](#)
  - ds\_str, [13](#)
  - ds\_vector, [14](#)
- data\_destructor
  - ds\_list, [8](#)
  - ds\_vector, [14](#)
- db\_connect
  - db\_connection.h, [19](#)
  - db\_dummy\_general.c, [40](#)
  - db\_mysql\_general.c, [45](#)
- db\_connection.h
  - db\_connect, [19](#)
- db\_create\_database\_structure
  - db\_structure.c, [31](#)
  - db\_structure.h, [33](#)
- db\_create\_entities\_table
  - db\_entities.c, [20](#)
  - db\_entities.h, [22](#)
- db\_create\_entities\_table\_sql
  - db\_dummy\_create\_entities\_table\_sql.c, [37](#)
  - db\_mysql\_create\_entities\_table\_sql.c, [42](#)
  - db\_sql.h, [29](#)
- db\_create\_recordset\_from\_query
  - db\_dummy\_general.c, [40](#)
  - db\_mysql\_general.c, [45](#)
  - db\_reporting.h, [26](#)
- db\_create\_report\_from\_query
  - db\_reporting.c, [26](#)
  - db\_reporting.h, [27](#)
- db\_create\_users\_table
  - db\_users.c, [34](#)
  - db\_users.h, [36](#)
- db\_create\_users\_table\_sql
  - db\_dummy\_create\_users\_table\_sql.c, [37](#)
  - db\_mysql\_create\_users\_table\_sql.c, [43](#)
  - db\_sql.h, [29](#)
- db\_delete\_database\_structure
  - db\_structure.c, [31](#)
  - db\_structure.h, [33](#)
- db\_drop\_entities\_table
  - db\_entities.c, [20](#)
  - db\_entities.h, [22](#)
- db\_drop\_entities\_table\_sql
  - db\_dummy\_drop\_entities\_table\_sql.c, [38](#)
  - db\_mysql\_drop\_entities\_table\_sql.c, [43](#)
  - db\_sql.h, [30](#)
- db\_drop\_users\_table
  - db\_users.c, [34](#)
  - db\_users.h, [36](#)
- db\_drop\_users\_table\_sql
  - db\_dummy\_drop\_users\_table\_sql.c, [38](#)
  - db\_mysql\_drop\_users\_table\_sql.c, [44](#)
  - db\_sql.h, [30](#)
- db\_dummy\_create\_entities\_table\_sql.c
  - db\_create\_entities\_table\_sql, [37](#)
- db\_dummy\_create\_users\_table\_sql.c
  - db\_create\_users\_table\_sql, [37](#)
- db\_dummy\_drop\_entities\_table\_sql.c
  - db\_drop\_entities\_table\_sql, [38](#)
- db\_dummy\_drop\_users\_table\_sql.c
  - db\_drop\_users\_table\_sql, [38](#)
- db\_dummy\_general.c
  - db\_connect, [40](#)
  - db\_create\_recordset\_from\_query, [40](#)
  - db\_execute\_query, [40](#)
- db\_dummy\_list\_entities\_report\_sql.c

- db\_list\_entities\_report\_sql, 41
- db\_dummy\_list\_users\_report\_sql.c
  - db\_list\_users\_report\_sql, 41
- db\_entities.c
  - db\_create\_entities\_table, 20
  - db\_drop\_entities\_table, 20
  - db\_list\_entities\_report, 20
- db\_entities.h
  - db\_create\_entities\_table, 22
  - db\_drop\_entities\_table, 22
  - db\_list\_entities\_report, 22
- db\_execute\_query
  - db\_dummy\_general.c, 40
  - db\_mysql\_general.c, 46
  - db\_query.h, 24
- db\_list\_entities\_report
  - db\_entities.c, 20
  - db\_entities.h, 22
- db\_list\_entities\_report\_sql
  - db\_dummy\_list\_entities\_report\_sql.c, 41
  - db\_mysql\_list\_entities\_report\_sql.c, 46
  - db\_sql.h, 30
- db\_list\_users\_report
  - db\_users.c, 34
  - db\_users.h, 36
- db\_list\_users\_report\_sql
  - db\_dummy\_list\_users\_report\_sql.c, 41
  - db\_mysql\_list\_users\_report\_sql.c, 47
  - db\_sql.h, 30
- db\_mysql\_create\_entities\_table\_sql.c
  - db\_create\_entities\_table\_sql, 42
- db\_mysql\_create\_users\_table\_sql.c
  - db\_create\_users\_table\_sql, 43
- db\_mysql\_drop\_entities\_table\_sql.c
  - db\_drop\_entities\_table\_sql, 43
- db\_mysql\_drop\_users\_table\_sql.c
  - db\_drop\_users\_table\_sql, 44
- db\_mysql\_general.c
  - conn\_mss, 46
  - db\_connect, 45
  - db\_create\_recordset\_from\_query, 45
  - db\_execute\_query, 46
  - main\_mss, 46
- db\_mysql\_list\_entities\_report\_sql.c
  - db\_list\_entities\_report\_sql, 46
- db\_mysql\_list\_users\_report\_sql.c
  - db\_list\_users\_report\_sql, 47
- db\_query.h
  - db\_execute\_query, 24
- db\_reporting.c
  - db\_create\_report\_from\_query, 26
- db\_reporting.h
  - db\_create\_recordset\_from\_query, 26
  - db\_create\_report\_from\_query, 27
- db\_sql.h
  - db\_create\_entities\_table\_sql, 29
  - db\_create\_users\_table\_sql, 29
  - db\_drop\_entities\_table\_sql, 30
  - db\_drop\_users\_table\_sql, 30
  - db\_list\_entities\_report\_sql, 30
  - db\_list\_users\_report\_sql, 30
- db\_structure.c
  - db\_create\_database\_structure, 31
  - db\_delete\_database\_structure, 31
- db\_structure.h
  - db\_create\_database\_structure, 33
  - db\_delete\_database\_structure, 33
- db\_users.c
  - db\_create\_users\_table, 34
  - db\_drop\_users\_table, 34
  - db\_list\_users\_report, 34
- db\_users.h
  - db\_create\_users\_table, 36
  - db\_drop\_users\_table, 36
  - db\_list\_users\_report, 36
- delim\_file\_read
  - delim\_file\_read.c, 104
  - delim\_file\_read.h, 106
- delim\_file\_read.c
  - delim\_file\_read, 104
  - MAX\_LINE\_SIZE, 104
- delim\_file\_read.h
  - delim\_file\_read, 106
- ds\_list, 7
  - current, 7
  - data\_destructor, 8
  - ds\_list.h, 52
  - free\_on\_delete, 8
  - head, 8
  - length, 8
  - tail, 8
- ds\_list.c
  - ds\_list\_create, 50
  - ds\_list\_destructor, 50
- ds\_list.h
  - ds\_list, 52
  - ds\_list\_append, 52
  - ds\_list\_create, 52
  - ds\_list\_destroy, 53
  - ds\_list\_destructor, 53
  - ds\_list\_element, 53
  - ds\_list\_get\_next\_data, 53
  - ds\_list\_get\_prev\_data, 54
  - ds\_list\_is\_empty, 54
  - ds\_list\_length, 54
  - ds\_list\_remove\_all, 54
  - ds\_list\_remove\_tail, 54
  - ds\_list\_seek\_end, 55
  - ds\_list\_seek\_start, 55
- ds\_list\_append
  - ds\_list.h, 52
- ds\_list\_create
  - ds\_list.c, 50
  - ds\_list.h, 52
- ds\_list\_destroy
  - ds\_list.h, 53

- ds\_list\_destructor
  - ds\_list.c, 50
  - ds\_list.h, 53
- ds\_list\_element, 8
  - data, 9
  - ds\_list.h, 53
  - next, 9
  - previous, 9
- ds\_list\_get\_next\_data
  - ds\_list.h, 53
- ds\_list\_get\_prev\_data
  - ds\_list.h, 54
- ds\_list\_is\_empty
  - ds\_list.h, 54
- ds\_list\_length
  - ds\_list.h, 54
- ds\_list\_remove\_all
  - ds\_list.h, 54
- ds\_list\_remove\_tail
  - ds\_list.h, 54
- ds\_list\_seek\_end
  - ds\_list.h, 55
- ds\_list\_seek\_start
  - ds\_list.h, 55
- ds\_map, 9
  - ds\_map.h, 58
  - hash\_size, 10
  - lists, 10
- ds\_map.c
  - ds\_map\_init, 57
  - ds\_map\_print\_all, 57
- ds\_map.h
  - ds\_map, 58
  - ds\_map\_destroy, 58
  - ds\_map\_get\_value, 58
  - ds\_map\_init, 59
  - ds\_map\_insert, 59
  - ds\_map\_print\_all, 59
- ds\_map\_destroy
  - ds\_map.h, 58
- ds\_map\_get\_value
  - ds\_map.h, 58
- ds\_map\_init
  - ds\_map.c, 57
  - ds\_map.h, 59
- ds\_map\_insert
  - ds\_map.h, 59
- ds\_map\_print\_all
  - ds\_map.c, 57
  - ds\_map.h, 59
- ds\_map\_str, 10
  - ds\_map\_str.h, 62
  - hash\_size, 10
  - lists, 11
- ds\_map\_str.c
  - ds\_map\_str\_init, 61
- ds\_map\_str.h
  - ds\_map\_str, 62
  - ds\_map\_str\_destroy, 62
  - ds\_map\_str\_get\_value, 62
  - ds\_map\_str\_init, 62
  - ds\_map\_str\_insert, 63
- ds\_map\_str\_destroy
  - ds\_map\_str.h, 62
- ds\_map\_str\_get\_value
  - ds\_map\_str.h, 62
- ds\_map\_str\_init
  - ds\_map\_str.c, 61
  - ds\_map\_str.h, 62
- ds\_map\_str\_insert
  - ds\_map\_str.h, 63
- ds\_record, 11
  - ds\_record.h, 67
  - fields, 11
- ds\_record.c
  - ds\_record\_create, 64
  - ds\_record\_destructor, 64
  - ds\_record\_make\_delim\_string, 65
  - ds\_record\_make\_values\_string, 65
  - ds\_record\_tokenize, 65
- ds\_record.h
  - ds\_record, 67
  - ds\_record\_clear, 67
  - ds\_record\_create, 67
  - ds\_record\_destroy, 67
  - ds\_record\_destructor, 68
  - ds\_record\_get\_field, 68
  - ds\_record\_get\_next\_data, 68
  - ds\_record\_make\_delim\_string, 68
  - ds\_record\_make\_values\_string, 69
  - ds\_record\_seek\_start, 69
  - ds\_record\_set\_field, 69
  - ds\_record\_size, 69
  - ds\_record\_tokenize, 69
- ds\_record\_clear
  - ds\_record.h, 67
- ds\_record\_create
  - ds\_record.c, 64
  - ds\_record.h, 67
- ds\_record\_destroy
  - ds\_record.h, 67
- ds\_record\_destructor
  - ds\_record.c, 64
  - ds\_record.h, 68
- ds\_record\_get\_field
  - ds\_record.h, 68
- ds\_record\_get\_next\_data
  - ds\_record.h, 68
- ds\_record\_make\_delim\_string
  - ds\_record.c, 65
  - ds\_record.h, 68
- ds\_record\_make\_values\_string
  - ds\_record.c, 65
  - ds\_record.h, 69
- ds\_record\_seek\_start
  - ds\_record.h, 69



- ds\_record\_set\_field
  - ds\_record.h, 69
- ds\_record\_size
  - ds\_record.h, 69
- ds\_record\_tokenize
  - ds\_record.c, 65
  - ds\_record.h, 69
- ds\_recordset, 12
  - ds\_recordset.h, 73
  - field\_lengths, 12
  - headers, 12
  - num\_fields, 12
  - records, 12
- ds\_recordset.c
  - ds\_recordset\_create, 71
- ds\_recordset.h
  - ds\_recordset, 73
  - ds\_recordset\_add\_record, 73
  - ds\_recordset\_create, 73
  - ds\_recordset\_destroy, 73
  - ds\_recordset\_get\_next\_insert\_query, 73
  - ds\_recordset\_get\_text\_report, 74
  - ds\_recordset\_next\_record, 74
  - ds\_recordset\_num\_fields, 74
  - ds\_recordset\_num\_records, 74
  - ds\_recordset\_seek\_start, 75
  - ds\_recordset\_set\_headers, 75
- ds\_recordset\_add\_record
  - ds\_recordset.h, 73
- ds\_recordset\_create
  - ds\_recordset.c, 71
  - ds\_recordset.h, 73
- ds\_recordset\_destroy
  - ds\_recordset.h, 73
- ds\_recordset\_get\_next\_insert\_query
  - ds\_recordset.h, 73
- ds\_recordset\_get\_text\_report
  - ds\_recordset.h, 74
- ds\_recordset\_next\_record
  - ds\_recordset.h, 74
- ds\_recordset\_num\_fields
  - ds\_recordset.h, 74
- ds\_recordset\_num\_records
  - ds\_recordset.h, 74
- ds\_recordset\_seek\_start
  - ds\_recordset.h, 75
- ds\_recordset\_set\_headers
  - ds\_recordset.h, 75
- ds\_str, 13
  - capacity, 13
  - data, 13
  - ds\_str.h, 86
  - length, 13
- ds\_str.c
  - ds\_str\_assign, 77
  - ds\_str\_assign\_cstr, 77
  - ds\_str\_char\_at\_index, 77
  - ds\_str\_clear, 78
  - ds\_str\_compare, 78
  - ds\_str\_compare\_cstr, 78
  - ds\_str\_concat, 78
  - ds\_str\_create, 79
  - ds\_str\_create\_direct, 79
  - ds\_str\_create\_sprintf, 79
  - ds\_str\_cstr, 79
  - ds\_str\_decorate, 80
  - ds\_str\_destroy, 80
  - ds\_str\_destructor, 80
  - ds\_str\_doubleval, 80
  - ds\_str\_dup, 81
  - ds\_str\_getline, 81
  - ds\_str\_intval, 81
  - ds\_str\_is\_empty, 81
  - ds\_str\_length, 82
  - ds\_str\_split, 82
  - ds\_str\_strchr, 82
  - ds\_str\_substr\_left, 82
  - ds\_str\_substr\_right, 83
  - ds\_str\_trim, 83
  - ds\_str\_trim\_leading, 83
  - ds\_str\_trim\_trailing, 83
  - ds\_str\_trunc, 83
- ds\_str.h
  - ds\_str, 86
  - ds\_str\_assign, 86
  - ds\_str\_assign\_cstr, 86
  - ds\_str\_char\_at\_index, 86
  - ds\_str\_clear, 87
  - ds\_str\_compare, 87
  - ds\_str\_compare\_cstr, 87
  - ds\_str\_concat, 87
  - ds\_str\_concat\_cstr, 87
  - ds\_str\_create, 88
  - ds\_str\_create\_direct, 88
  - ds\_str\_create\_sprintf, 88
  - ds\_str\_cstr, 89
  - ds\_str\_decorate, 89
  - ds\_str\_destroy, 89
  - ds\_str\_destructor, 89
  - ds\_str\_doubleval, 89
  - ds\_str\_dup, 90
  - ds\_str\_getline, 90
  - ds\_str\_hash, 90
  - ds\_str\_intval, 90
  - ds\_str\_is\_empty, 91
  - ds\_str\_length, 91
  - ds\_str\_split, 91
  - ds\_str\_strchr, 91
  - ds\_str\_substr\_left, 92
  - ds\_str\_substr\_right, 92
  - ds\_str\_trim, 92
  - ds\_str\_trim\_leading, 92
  - ds\_str\_trim\_trailing, 93
  - ds\_str\_trunc, 93
- ds\_str\_assign
  - ds\_str.c, 77

- ds\_str.h, [86](#)
- ds\_str\_assign\_cstr
  - ds\_str.c, [77](#)
  - ds\_str.h, [86](#)
- ds\_str\_char\_at\_index
  - ds\_str.c, [77](#)
  - ds\_str.h, [86](#)
- ds\_str\_clear
  - ds\_str.c, [78](#)
  - ds\_str.h, [87](#)
- ds\_str\_compare
  - ds\_str.c, [78](#)
  - ds\_str.h, [87](#)
- ds\_str\_compare\_cstr
  - ds\_str.c, [78](#)
  - ds\_str.h, [87](#)
- ds\_str\_concat
  - ds\_str.c, [78](#)
  - ds\_str.h, [87](#)
- ds\_str\_concat\_cstr
  - ds\_str.h, [87](#)
- ds\_str\_create
  - ds\_str.c, [79](#)
  - ds\_str.h, [88](#)
- ds\_str\_create\_direct
  - ds\_str.c, [79](#)
  - ds\_str.h, [88](#)
- ds\_str\_create\_sprintf
  - ds\_str.c, [79](#)
  - ds\_str.h, [88](#)
- ds\_str\_cstr
  - ds\_str.c, [79](#)
  - ds\_str.h, [89](#)
- ds\_str\_decorate
  - ds\_str.c, [80](#)
  - ds\_str.h, [89](#)
- ds\_str\_destroy
  - ds\_str.c, [80](#)
  - ds\_str.h, [89](#)
- ds\_str\_destructor
  - ds\_str.c, [80](#)
  - ds\_str.h, [89](#)
- ds\_str\_doubleval
  - ds\_str.c, [80](#)
  - ds\_str.h, [89](#)
- ds\_str\_dup
  - ds\_str.c, [81](#)
  - ds\_str.h, [90](#)
- ds\_str\_getline
  - ds\_str.c, [81](#)
  - ds\_str.h, [90](#)
- ds\_str\_hash
  - ds\_str.h, [90](#)
- ds\_str\_intval
  - ds\_str.c, [81](#)
  - ds\_str.h, [90](#)
- ds\_str\_is\_empty
  - ds\_str.c, [81](#)
- ds\_str.h, [91](#)
- ds\_str\_length
  - ds\_str.c, [82](#)
  - ds\_str.h, [91](#)
- ds\_str\_split
  - ds\_str.c, [82](#)
  - ds\_str.h, [91](#)
- ds\_str\_strchr
  - ds\_str.c, [82](#)
  - ds\_str.h, [91](#)
- ds\_str\_substr\_left
  - ds\_str.c, [82](#)
  - ds\_str.h, [92](#)
- ds\_str\_substr\_right
  - ds\_str.c, [83](#)
  - ds\_str.h, [92](#)
- ds\_str\_trim
  - ds\_str.c, [83](#)
  - ds\_str.h, [92](#)
- ds\_str\_trim\_leading
  - ds\_str.c, [83](#)
  - ds\_str.h, [92](#)
- ds\_str\_trim\_trailing
  - ds\_str.c, [83](#)
  - ds\_str.h, [93](#)
- ds\_str\_trunc
  - ds\_str.c, [83](#)
  - ds\_str.h, [93](#)
- ds\_vector, [13](#)
  - current, [14](#)
  - data, [14](#)
  - data\_destructor, [14](#)
  - ds\_vector.h, [96](#)
  - free\_on\_delete, [14](#)
  - size, [14](#)
- ds\_vector.c
  - ds\_vector\_create, [94](#)
  - ds\_vector\_destructor, [94](#)
- ds\_vector.h
  - ds\_vector, [96](#)
  - ds\_vector\_clear, [96](#)
  - ds\_vector\_create, [96](#)
  - ds\_vector\_destroy, [97](#)
  - ds\_vector\_destructor, [97](#)
  - ds\_vector\_element, [97](#)
  - ds\_vector\_get\_next\_data, [97](#)
  - ds\_vector\_seek\_start, [98](#)
  - ds\_vector\_set, [98](#)
  - ds\_vector\_size, [98](#)
- ds\_vector\_clear
  - ds\_vector.h, [96](#)
- ds\_vector\_create
  - ds\_vector.c, [94](#)
  - ds\_vector.h, [96](#)
- ds\_vector\_destroy
  - ds\_vector.h, [97](#)
- ds\_vector\_destructor
  - ds\_vector.c, [94](#)

- ds\_vector.h, [97](#)
- ds\_vector\_element
  - ds\_vector.h, [97](#)
- ds\_vector\_get\_next\_data
  - ds\_vector.h, [97](#)
- ds\_vector\_seek\_start
  - ds\_vector.h, [98](#)
- ds\_vector\_set
  - ds\_vector.h, [98](#)
- ds\_vector\_size
  - ds\_vector.h, [98](#)
- field\_lengths
  - ds\_recordset, [12](#)
- fields
  - ds\_record, [11](#)
- free\_on\_delete
  - ds\_list, [8](#)
  - ds\_vector, [14](#)
- gl\_log\_msg
  - gl\_logging.c, [108](#)
  - gl\_logging.h, [110](#)
- gl\_logging.c
  - gl\_log\_msg, [108](#)
  - gl\_set\_logging, [109](#)
- gl\_logging.h
  - gl\_log\_msg, [110](#)
  - gl\_set\_logging, [110](#)
- gl\_set\_logging
  - gl\_logging.c, [109](#)
  - gl\_logging.h, [110](#)
- hash\_size
  - ds\_map, [10](#)
  - ds\_map\_str, [10](#)
- head
  - ds\_list, [8](#)
- headers
  - ds\_recordset, [12](#)
- key
  - kv\_pair\_node, [15](#)
- kv\_pair\_node, [14](#)
  - key, [15](#)
  - next, [15](#)
  - value, [15](#)
- length
  - ds\_list, [8](#)
  - ds\_str, [13](#)
- lib/database/database.h, [17](#)
- lib/database/db\_connection.h, [18](#)
- lib/database/db\_entities.c, [19](#)
- lib/database/db\_entities.h, [21](#)
- lib/database/db\_internal.h, [23](#)
- lib/database/db\_query.h, [23](#)
- lib/database/db\_reporting.c, [25](#)
- lib/database/db\_reporting.h, [26](#)
- lib/database/db\_sampledata.c, [27](#)
- lib/database/db\_sampledata.h, [28](#)
- lib/database/db\_sql.h, [29](#)
- lib/database/db\_structure.c, [30](#)
- lib/database/db\_structure.h, [32](#)
- lib/database/db\_users.c, [33](#)
- lib/database/db\_users.h, [35](#)
- lib/database/dummy/db\_dummy\_create\_entities\_table\_sql.c, [36](#)
- lib/database/dummy/db\_dummy\_create\_users\_table\_sql.c, [37](#)
- lib/database/dummy/db\_dummy\_drop\_entities\_table\_sql.c, [37](#)
- lib/database/dummy/db\_dummy\_drop\_users\_table\_sql.c, [38](#)
- lib/database/dummy/db\_dummy\_general.c, [39](#)
- lib/database/dummy/db\_dummy\_list\_entities\_report\_sql.c, [40](#)
- lib/database/dummy/db\_dummy\_list\_users\_report\_sql.c, [41](#)
- lib/database/mysql/db\_mysql\_create\_entities\_table\_sql.c, [42](#)
- lib/database/mysql/db\_mysql\_create\_users\_table\_sql.c, [42](#)
- lib/database/mysql/db\_mysql\_drop\_entities\_table\_sql.c, [43](#)
- lib/database/mysql/db\_mysql\_drop\_users\_table\_sql.c, [43](#)
- lib/database/mysql/db\_mysql\_general.c, [44](#)
- lib/database/mysql/db\_mysql\_list\_entities\_report\_sql.c, [46](#)
- lib/database/mysql/db\_mysql\_list\_users\_report\_sql.c, [47](#)
- lib/datastruct/data\_structures.h, [47](#)
- lib/datastruct/ds\_list.c, [48](#)
- lib/datastruct/ds\_list.h, [50](#)
- lib/datastruct/ds\_map.c, [55](#)
- lib/datastruct/ds\_map.h, [57](#)
- lib/datastruct/ds\_map\_str.c, [59](#)
- lib/datastruct/ds\_map\_str.h, [61](#)
- lib/datastruct/ds\_record.c, [63](#)
- lib/datastruct/ds\_record.h, [65](#)
- lib/datastruct/ds\_recordset.c, [70](#)
- lib/datastruct/ds\_recordset.h, [71](#)
- lib/datastruct/ds\_str.c, [75](#)
- lib/datastruct/ds\_str.h, [84](#)
- lib/datastruct/ds\_vector.c, [93](#)
- lib/datastruct/ds\_vector.h, [95](#)
- lib/file\_ops/config\_file\_read.c, [98](#)
- lib/file\_ops/config\_file\_read.h, [100](#)
- lib/file\_ops/delim\_file\_read.c, [103](#)
- lib/file\_ops/delim\_file\_read.h, [104](#)
- lib/file\_ops/file\_ops.h, [106](#)
- lib/gl\_general/gl\_logging.c, [108](#)
- lib/gl\_general/gl\_logging.h, [109](#)
- lists
  - ds\_map, [10](#)
  - ds\_map\_str, [11](#)

- MAX\_BUFFER\_SIZE
  - config\_file\_read.c, [99](#)
- MAX\_LINE\_SIZE
  - delim\_file\_read.c, [104](#)
- main
  - main.c, [111](#)
- main.c, [110](#)
  - main, [111](#)
- main\_mss
  - db\_mysql\_general.c, [46](#)
- next
  - ds\_list\_element, [9](#)
  - kv\_pair\_node, [15](#)
- num\_fields
  - ds\_recordset, [12](#)
- params, [16](#)
- previous
  - ds\_list\_element, [9](#)
- records
  - ds\_recordset, [12](#)
- size
  - ds\_vector, [14](#)
- tail
  - ds\_list, [8](#)
- value
  - kv\_pair\_node, [15](#)